KAMINO: Constraint-Aware Differentially Private Data Synthesis

Chang Ge, Shubhankar Mohapatra, Xi He, Ihab F. Ilyas
University of Waterloo
{c4ge,s3mohapatra,xihe,ilyas}@uwaterloo.ca

ABSTRACT

Organizations are increasingly relying on data to support decisions. When data contains private and sensitive information, the data owner often desires to publish a synthetic database instance that is similarly useful as the true data, while ensuring the privacy of individual data records. Existing differentially private data synthesis methods aim to generate useful data based on applications, but they fail in keeping one of the most fundamental data properties of the structured data — the underlying correlations and dependencies among tuples and attributes (i.e., the structure of the data). This structure is often expressed as integrity and schema constraints, or with a probabilistic generative process. As a result, the synthesized data is not useful for any downstream tasks that require this structure to be preserved.

This work presents KAMINO, a data synthesis system to ensure differential privacy and to preserve the structure and correlations present in the original dataset. KAMINO takes as input of a database instance, along with its schema (including integrity constraints), and produces a synthetic database instance with differential privacy and structure preservation guarantees. We empirically show that while preserving the structure of the data, KAMINO achieves comparable and even better usefulness in applications of training classification models and answering marginal queries than the state-of-the-art methods of differentially private data synthesis.

PVLDB Reference Format:
Chang Ge, Shubhankar Mohapatra, Xi He, Ihab F. Ilyas. KAMINO: Constraint-Aware Differentially Private Data Synthesis. PVLDB, 14(10): 1886 - 1899, 2021.
doi:10.14778/3467861.3467876

1 INTRODUCTION

Organizations have been extensively relying on personal data to support a growing spectrum of businesses, from music recommendations to life-saving coronavirus research [77]. This type of data is often structured and carries sensitive information about individuals. Reckless data sharing for data-driven applications and research causes great privacy concerns [10, 47] and penalties [1]. Differential privacy (DP) [28] has emerged as a standard data privacy guarantee by government agencies [9, 45] and companies [34, 43, 49]. Informally, the output of a data sharing process that satisfies DP has a similar distribution whether an individual's data is used for the computation or not. Hence, the output cannot be used to infer much about any individual's data and therefore is considered "private".

Proceedings of the VLDB Endowment, Vol. 14, No. 10 ISSN 2150-8097.
doi:10.14778/3467861.3467876

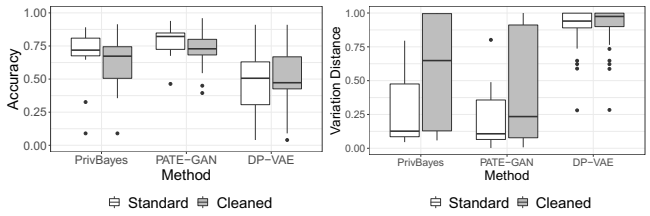
Differential privacy is often achieved via randomization, such as injecting controlled noise into the input data [54] based on the required privacy level, and hence there is a trade-off between privacy and the utility of this data to downstream applications. One approach often followed in prior work focuses on the optimization of this trade-off for a given application (e.g., releasing statistics [9, 18], building prediction models [8, 63], answering SQL queries [40, 49, 53, 58]). For example, APEx [40] is designed for data exploration; for each query, APEx searches the best differentially private algorithm that can answer the query accurately with the minimum privacy cost. This line of work allows the fine-tuning of an algorithm for the optimal trade-off between the privacy cost and the accuracy of the given application, but the released output may not be useful for other applications. Running a new application on the same dataset usually requires additional privacy cost.

An attractive alternative approach is to publish a differentially private synthetic database instance with a set of desired properties such as similar value distributions or dependency structure, with the hope that it has the same utility or is as useful as the original dataset to a large class of downstream applications that require those properties. For example, the US Census Bureau released differentially private census data, and it has been shown useful to keep similar home-workplace distribution as the true data to populate the mapping application [18]. Privately releasing synthetic data avoids designing separate mechanism for each target application, and the privacy cost is incurred only once for all supported applications due to the post-processing property of differential privacy [32].

1.1 Problems with Current DP Data Synthesis

For applications that consume structured data with predefined schema in a SQL database, it is important for the synthetic data to keep *the structure of the data* — the underlying correlations and dependencies among tuples and attributes. This structure is often expressed as integrity and schema constraints, such as functional dependencies between attributes or key constraints between tables. Otherwise, the synthesized data is not useful for any downstream tasks that require this structure to be preserved.

In general, generating differentially private synthetic data based on true data faces fundamental challenges. Take answering statistical queries as an example application. Prior work [16, 31, 39, 75] have shown that the running time for sampling a synthetic dataset that is accurate for answering a large family of statistics (e.g., all α-way marginals) grows exponentially in the dimension of the data. On the other hand, an efficient private data generation algorithm fails to offer the same level of accuracy guarantees to all the queries. Existing practical methods (e.g., [12, 20, 21, 50, 83]) therefore choose to privately learn only a subset of queries or correlations to model the true data and then sample database instances based on the learned information. However, the structure of the data is not explicitly captured by these methods and thus are poorly preserved in



(a) Accuracy **(b) 2-way marginal**

Figure 1: A synthetic Adult data using PrivBayes, PATE-GAN and DP-VAE satisfying ($\epsilon = 1, \delta = 10^{-6}$)-DP, with and without fixing the integrity violations (labeled as 'cleaned' and 'standard', respectively). Each point in Figure 1a represents the testing accuracy for one target attribute. Each point in Figure 1b represents the total variation distance between the true and synthetic Adult. More details are in § 7.

the synthetic data. In particular, all these methods assume tuples in the database instance are independent and identically distributed (i.i.d.), and sample each tuple independently. The output database instance has a significant number of violations to the structure constraints in the truth.

Example 1: Consider the Adult dataset [27] consisting of 15 attributes with denial constraints [48], such as 'two tuples with the same education category cannot have different education numbers', and 'tuples with higher capital gain cannot have lower capital loss'. There is no single violation of these constraints in the true data, but the synthetic data generated by the state-of-the-arts including PrivBayes [83], PATE-GAN [50], and DP-VAE [20] have up to 32% of the tuple pairs failing these constraints (Table 2).

However, naïvly repairing the incorrect structure constraints in the synthetic data can compromise the usefulness. We applied state-of-the-art data cleaning method [67] to fix the violations in the synthetic data generated by the aforementioned three methods. Then we evaluated their usefulness in training classification models and building 2-way marginals. Figure 1 shows that the repaired synthetic data (labeled as 'cleaned') have lower classification quality (i.e., smaller accuracy score) and poorer marginals (i.e., larger distance) compared to the synthetic data with violations (labeled as 'standard'). Though the repaired synthetic data managed to comply with structure constraints, they become less useful for training models and releasing marginal statistics. □

1.2 Constraint-Aware DP Data Synthesis

To solve the aforementioned problems, we are motivated to design an end-to-end synthetic data generator that preserves both the structure of the data and the privacy of individual data records. In this work, we consider an important class of structure constraints, the *denial constraints* (DCs) [48], and we present Kamino[1], a system for constraint-aware differentially private data synthesis.

Our solution is built on top of the probabilistic database framework [69, 72], which models a probability distribution over ordinary databases and incorporates the denial constrains as parametric factors. Database instances that share similar structural and statistical correlations with the true data are modeled to have similar probabilities. We first privately learn a parametric model of the probabilistic database, and then sample a database instance from the model as a post-processing step. To make it more efficient, we decompose the joint probability of a database instance into a chain of conditional probabilities, and privately estimate tuple distribution using tuple embedding [79] and the attention mechanism [11] for mixed data types (categorical and numerical). As we explicitly consider additional correlation structures compared to prior work, Kamino can incur more performance cost and utility cost for other applications given the same level of privacy constraint. Our empirical evaluation shows that the performance overhead and accuracy payoff are negligible. We also show that while preserving DCs, Kamino produces synthetic data that have comparable and even better quality for classification applications and marginal queries than the state-of-the-art methods on differentially private data synthesis.

We highlight the main contributions of this work as follows:

- We believe this is the first work to consider denial constraints in differentially private data synthesis, which are important properties for structured data. We use probabilistic database framework to incorporate DCs and attribute correlations.
- We develop an efficient learning and sampling algorithm for Kamino by decomposing the probabilistic database model into a chain of submodels, based on the given constraints (§ 3 & § 4).
- We design a private learning algorithm in Kamino to learn the weights of given DCs to allow interpreting them in the model as soft constraints (§ 5).
- We build the prototype for Kamino, the first end-to-end system for differentially private data generation with DCs, and apply advanced privacy composition techniques to obtain a tight end-to-end privacy bound (§ 6).

We evaluate Kamino over real-world datasets and show that the synthetic data have similar violations to the given DCs as in the true data, and they also achieve the best or close to best data usefulness in the marginal queries (variation distance) and the learning tasks (accuracy and F1), compared to the state-of-the-art methods (§ 7).

2 PRELIMINARIES

We consider a relational database schema of a single relation $R = \{A_1, \cdots, A_k\}$ with k attributes. Let D be a database instance of this schema R and consist of n tuples $\{t_1, \cdots, t_n\}$. Each tuple $t_i \in D$ has an implicit identifier i, and $t_i[A_j]$ denotes the value taken by the tuple t_i for attribute A_j from its domain $\mathcal{D}(A_j)$. Index 1 refers to the first element in a list/array.

2.1 Denial Constraints

Denial constraints (DCs) [48] are used in practice by domain experts to specify the structure of the data, such as functional dependency (FD) [46] and conditional FD [37]. In case of missing DCs, recent work have designed algorithms to automatically discover DCs from the database instance [15, 24].

We express a DC as a first-order formula in the form of $\phi : \forall t_i, t_j, \cdots \in D, \neg(P_1 \wedge \cdots \wedge P_m)$. Each predict P_i is of the form $(v_1\ o\ v_2)$ or $(v_1\ o\ c)$, where $v_1, v_2 \in t_x[A]$, $x \in \{i, j, \cdots\}$, $A \in R$,

[1]Kamino was the planet in Star Wars, renowned for the technology of clone armies.

$o \in \{=, \neq, >, \geq, <, \leq\}$, and c is a constant. We will omit universal quantifiers $\forall t_i, t_j, \ldots$ hereafter for simplicity.

Example 2: Consider a database instance D with schema $R = \{age, edu\_num, edu, cap\_gain, cap\_loss\}$, and three DCs:
ϕ_1: $\neg(t_i[edu] = t_j[edu] \wedge t_i[edu\_num] \neq t_j[edu\_num])$
ϕ_2: $\neg(t_i[cap\_gain] > t_j[cap\_gain] \wedge t_i[cap\_loss] < t_j[cap\_loss])$
ϕ_3: $\neg(t_i[age] < 10 \wedge t_i[cap\_gain] > 1M)$

The first DC ϕ_1 expresses an FD $edu \rightarrow edu\_num$. It states that for any two tuples with same *edu*, their *edu_num* must be the same too. The second DC ϕ_2 states that for any two tuples, if one's *cap_gain* is greater than the other's, its *cap_loss* cannot be smaller. The third DC ϕ_3 is a unary DC that enforces every tuple with *age* less than 10 cannot have *cap_gain* more than 1 million. □

A DC states that all the predicts cannot be true at the same time, otherwise, a violation occurs. We use $V(\phi, D)$ to represent the set of tuples (for unary DCs) or tuple groups (for non-unary DCs) that violates DC ϕ in a database instance D. we refer to DC ϕ as a hard DC if no violations are allowed (i.e., $V(\phi, D) = \emptyset$), or a soft DC if a database instance can have violations. Note that the set of DC violations expands monotonicity with respect to the size of a database instance, that is for a subset instance $\hat{D} \subset D$, $V(\phi, \hat{D}) \subset V(\phi, D)$. We also use $\mathcal{A}_\phi$ to represent the set of attributes that participate in the DC ϕ. For example, $\mathcal{A}_{\phi_1} = \{edu, edu\_num\}$.

2.2 Probabilistic Database

The probabilistic database framework [69] has been used in practice [67, 79] to model observed data that do not fully comply with a given set of DCs. Intuitively, a database instance with few violations is more likely. Given a set of DCs Φ and their weights $\{w_\phi \mid \phi \in \Phi\}$, the probability of an instance D is defined as follows:

$$\Pr(D) \propto \prod_{t \in D} \Pr(t) \times \exp(-\sum_{\phi \in \Phi} w_\phi \times |V(\phi, D)|) \quad (1)$$

where $\prod_{t \in D} \Pr(t)$ models a tuple-independent probabilistic database [69, 72], wherein each tuple independently comes from a probability distribution over tuples, and $|V(\phi, D)|$ is the size of violations of DC ϕ on D. Each DC ϕ is associated with a weight w_ϕ and each violation of ϕ contributes a factor of $\exp(-w_\phi)$ to the probability of a random database instance D. This model captures both hard and soft DCs. For hard DCs, we set weights to be infinitely large, then a database instance with any violations has a small probability. For soft DCs, having more violations decreases its probability.

To learn a probabilistic database, one needs to learn the probability of tuples $\Pr(t)$ as well as the weights of DCs w_ϕ. The goal is to find the set of parameters $\{\Pr(t), w_\phi\}$ that maximizes the product of the likelihoods of all the training database samples [69]. The observed data will be used to learn the parameters in the model. We assume the distribution does not change.

2.3 Tuple Embedding

In this work, we express the tuple probability as the product of a chain of conditional probabilities:

$$\Pr(t) = \Pr(t[A_1]) \prod_{j=2}^{k} \Pr(t[A_j] \mid t[A_1, \cdots, A_{j-1}]) \quad (2)$$

Each conditional probability is learned as a discriminative model based on *tuple embedding* [79] and *attention mechanism* [11]. Similar to word embedding that models words in vectors of real numbers [59], tuple embedding has been applied to model tuples by encoding tuples into the space of real numbers [33, 79].

2.4 Differential Privacy

Differential privacy (DP) [30, 32] is used as our measure of privacy.

Definition 1 (Differential Privacy (DP) [32]). *A randomized algorithm $\mathcal{M}$ achieves (ϵ, δ)-DP if for all $\mathcal{S} \subseteq Range(\mathcal{M})$ and for any two database instances $D, D' \in \mathcal{D}$ that differ only in one tuple:*

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^\epsilon \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta.$$

The privacy cost is measured by the parameters (ϵ, δ). The smaller the privacy parameters are, the stronger the privacy offers. Complex DP algorithms can be built from the basic algorithms following two important properties of differential privacy: 1) Post-processing [29] states that for any function g defined over the output of the mechanism $\mathcal{M}$, if $\mathcal{M}$ satisfies (ϵ, δ)-DP, so does $g(\mathcal{M})$; 2) Composability [28] states that if $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_k$ satisfy (ϵ_1, δ_1)-, (ϵ_1, δ_1)-, $\cdots$, (ϵ_k, δ_k)-DP, then a mechanism sequentially applying $\mathcal{M}_1$, $\mathcal{M}_2$ to $\mathcal{M}_k$ satisfies $(\sum_{i=1}^k \epsilon_i, \sum_{i=1}^k \delta_i)$-DP.

Gaussian mechanism [32] is a widely used DP algorithm. Given a function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the Gaussian mechanism adds noise sampled from a Gaussian distribution $\mathcal{N}(0, S_f^2 \sigma^2)$ to each component of the query output, where σ is the noise scale and S_f is the L_2 sensitivity of function f, which is defined as $S_f = \max_{D, D' \text{differ in a row}} ||f(D) - f(D')||_2$. For $\epsilon \in (0, 1)$, if $\sigma \geq \sqrt{2\ln(1.25/\delta)}/\epsilon$, then the Gaussian mechanism satisfies (ϵ, δ)-DP.

Gaussian mechanism has been applied to answer counting queries [55]. It has also been used in differentially private stochastic gradient descent (DPSGD) [8, 13, 71, 78]. The gradients of SGD are the random variables to which the noise is added. As there is no a priori bound of the gradient, the sensitivity S_f is set by clipping the maximum L_2 norm of the gradient to a user-defined parameter C.

3 KAMINO OVERVIEW

To solve the shortcomings of the current differentially private data synthesis approaches mentioned in § 1.1, we state our problem definition and provide a high-level description of our approach.

3.1 Problem Statement

Given a private database instance D^* with schema and domain, a set of denial constraints Φ with information about their hardness, and a differential privacy budget (ϵ, δ), we would like to design a process P that generates a useful synthetic database instance D' as D^* (e.g., the same statistics and attribute correlations) while meeting two additional requirements:

R1. (Data Consistency) We consider data consistency with respect to the set of denial constraints Φ from the input: for each DC $\phi \in \Phi$, D^* and D' have a similar number of violations, i.e., $|V(\phi, D')| \approx |V(\phi, D^*)|$.

R2. (Privacy Guarantee) The process P that outputs D' achieves (ϵ, δ)-differential privacy: for any set of output instances $\mathbb{D}$

outputted by P, $\Pr(P(D_1) \in \mathbb{D}) \leq e^{\epsilon} \Pr(P(D_2) \in \mathbb{D}) + \delta$, for any two neighboring D_1 and D_2 differing in one record.

DC constraints Φ are public in our problem and can be modeled as part of the adversary's prior. This subsumes the special case when Φ are not public to the adversary. More discussion on the semantics of DP can be found in our full paper [41].

3.2 Methodology Overview

Recall from § 2.2, the probabilistic database model is a parametric model to describe the probability of instances. We adopt the probabilistic database model to represent databases with denial constraints. There are two main steps: (i) privately learn the unknown parameters in the probabilistic database model with samples from the true data; (ii) sample a database instance based on the learned probabilistic database model. However, both steps are challenging. First, it is well known that finding the analytical solution of the parameters of a probabilistic database without privacy concerns is #P-complete [68, 72], and approximate methods such as gradient descent may not converge to a global optimum [69], due to the large sampling space of tuples (cross product of all attributes' domain sizes) and of instances (exponential to the number of possible tuples). Second, prior work [16, 31, 39, 75] show that there is no efficient DP algorithm that can generate a database, which maintains accurate answers for an exponential family of learning concepts (e.g. the set of parameters in the probabilistic database model).

To tackle both the efficiency and the privacy challenge, we factorize the probability distribution of a database instance into a set of conditional probabilities given a subset of tuples and attributes, and learn them accordingly. We sample an instance based on the learned conditional probabilities.

Probabilistic database decomposition. We express the probability distribution of a database instance in Eqn. (1) into a chain of conditional probabilities based on two sequences (i) a sequence of tuple ids; and (ii) a sequence of attributes.

First, given a sequence of tuple ids $(1, 2, \ldots, n)$ in D, for any DC ϕ, the set of its violations in D, i.e., $V(\phi, D)$, can be iteratively computed by adding new violations introduced by tuple t_i with respect to its prefix tuples $D_{:i} = [t_1, t_2, \cdots, t_{i-1}]$ (with $D_{:1} = \emptyset$) from D, for $i = 1, \ldots, n$. Let $V(\phi, t_i \mid D_{:i})$ denote the set of new violations caused by tuple t_i with respect to $D_{:i}$. Then we have

$$|V(\phi, D)| = |V(\phi, t_1)| + |V(\phi, t_2 \mid D_{:2})| + \cdots + |V(\phi, t_n \mid D_{:n})|$$
$$= \sum_{i=1}^{n} |V(\phi, t_i \mid D_{:i})| \tag{3}$$

This allows us to decompose Eqn. (1) as

$$\Pr(D) \propto \left(\prod_{i=1}^{n} \Pr(t_i)\right) \times \exp\left(-\sum_{\phi \in \Phi} w_\phi \sum_{i=1}^{n} |V(\phi, t_i \mid D_{:i})|\right)$$
$$= \prod_{i=1}^{n} \left[\Pr(t_i) \times \exp\left(-\sum_{\phi \in \Phi} w_\phi \times |V(\phi, t_i \mid D_{:i})|\right)\right] \tag{4}$$

Next, we define a schema sequence S as an ordered list of all attributes in the schema. Similarly, let $S_{:j}$ represent all prefix attributes of the jth attribute in S and $S_{:1} = \emptyset$ for the purpose of uniform representation. This schema sequence allows us further

	age	edu_num	edu	cap_gain	cap_loss
t_1	39	13	Bachelors		
t_2	50	13	?		
t_3	38	9			
t_4	42	10			

Domain value	Cond. Pro.	#Vios	Sampling Pro.
Bachelors	0.3	0	1
HS-grad	0.3	1	0
Some-college	0.4	1	0

Figure 2: Sampling values in an instance (Example 4).

decompose the set of violations. Let Φ_{A_j} represent the set of DCs in Φ that can be fully expressed with the first j attributes in S, but cannot be expressed with only the first $j-1$ attributes.

Example 3: Continue with Example 2, given a schema sequence $S = [age, edu\_num, edu, cap\_gain, cap\_loss]$, we can verify that $\Phi_{A_3} = \{\phi_1\}$, as the attributes $\{edu\_num, edu\}$ for ϕ_1 are covered by the first 3 attributes in S, but not the first 2 attributes. □

Notice that for a DC $\phi \in \Phi_{A_j}$, given a tuple t_i, its number of violations $|V(\phi, t_i \mid D_{:i})|$ only depends on the values of the first j attributes in S (i.e., $S_{:j+1}$). As a result, we can rewrite the weighted sum of violations from Eqn. (4) as follows:

$$\sum_{\phi \in \Phi} w_\phi \times |V(\phi, t_i \mid D_{:i})| = \sum_{j=1}^{k} \sum_{\phi \in \Phi_{A_j}} w_\phi \times |V(\phi, t_i \mid D_{:i})|$$
$$= \sum_{j=1}^{k} \sum_{\phi \in \Phi_{A_j}} w_\phi \times |V(\phi, t_i[S_{:j+1}] \mid D_{:i}[S_{:j+1}])| \tag{5}$$

Based on the same schema sequence S, the tuple probability $\Pr[t_i]$ can be written as $\prod_{j=1}^{k} \Pr\left(t_i[A_j] \mid t_i[S_{:j}]\right)$ by the chain rule.

Finally, we have the database probability in Eqn. (4) expressed as

$$\Pr(D) \propto \prod_{j=1}^{k} \prod_{i=1}^{n} \Big[\Pr(t_i[A_j] \mid t_i[S_{:j+1}]) \times \exp(-\sum_{\phi \in \Phi_{A_j}} w_\phi \times |V(\phi, t_i[S_{:j+1}] \mid D_{:i}[S_{:j+1}])|)\Big] \tag{6}$$

Eqn. (6) in fact presents an iterative process to sample a database instance D based on (i) the schema sequence ($j \in [1, k]$), and (ii) the tuple id sequence ($i \in [1, n]$). Unlike the tuple id sequence, the schema sequence specifies an ordering of attributes, where each attribute solely depends on the prefix attributes to make correct prediction. However, it is challenging to find the optimal schema sequence [23], and hence we apply a greedy heuristic algorithm to derive a good one. In this work, we assume $\Pr[t_i]$ are the same for all tuples. Therefore, we just need to learn k (conditional) probabilities $\Pr(t[A_j] | t[S_{:j+1}])$, the weight of DCs w_ϕ, and the number of DC violations with respect to the prefix tuples. We will use the following example to illustrate the sampling process.

Example 4: Continue with Examples 2 and 3. Consider all three DCs be hard with infinitely large weight w_ϕ. Suppose we have already privately learned the conditional distributions from the true data. The construction of D' of 4 tuples works as follows.

We start with the first attribute *age*. From t_1 to t_4, we sample a value independently based on the distribution $\Pr(t[age])$. Then, we move on to the second attribute, *edu_num*. There is no DC between

Algorithm 1 Constraint-aware differentially private data synthesis

Require: Private instance D^*, schema R, domain $\mathcal{D}$
Require: DCs Φ, privacy budget (ϵ, δ)
1: **procedure** KAMINO($D^*, R, \mathcal{D}, \Phi, \epsilon, \delta$)
2: $S \leftarrow$ SEQUENCING($R, \mathcal{D}, \Phi$) ▷ Algorithm 4
3: $\Psi \leftarrow$ SEARCHDPARAS($\epsilon, \delta, \mathcal{D}, S$) ▷ Algorithm 6
4: $M \leftarrow$ TRAINMODEL($D^*, S, \mathcal{D}, \Psi$) ▷ Algorithm 2
5: $W \leftarrow$ LEARNWEIGHT(D^*, Φ, S, M, Ψ) ▷ Algorithm 5
6: $D' \leftarrow$ SYNTHESIZE($S, M, \Phi, \mathcal{D}, W$) ▷ Algorithm 3
7: **return** D'
8: **end procedure**

age and *edu_num*, each cell from t_1 to t_4 is filled with a sample based on the conditional distribution $\Pr(t[edu\_num] \mid t[age])$.

Next, for the third attribute *edu* (shown in Figure 2), DC ϕ_1 becomes active as all its relevant attributes (*edu_num*, *edu*) have been seen in the sequence. A cell value `Bachelors` is directly sampled for $t_1[edu]$ from the the conditional distribution $\Pr(t[edu] \mid t[edu\_num = 13, age = 39])$. For $t_2[edu]$, let's say the noisy conditional distributions of *edu* given *age* = 50 and *edu_num* = 13 are: (`Bachelors`, 0.3), (`HS-grad`, 0.3) and (`Some-college`, 0.4). Consider the infinitely large weight for ϕ_1, *edu* values other than `Bachelors` will cause violations to t_1 and hence their probabilities become very small. Therefore, `Bachelors` is sampled with high probability.

After all cells are filled, we get a synthetic instance D'. Optionally, the Markov Chain Monte Carlo (MCMC) sampling [62] could be applied to improve the accuracy by randomly choosing a cell $t_i[A_j]$ to re-sample, conditioning on all other cells $D' \setminus \{t_i[A_j]\}$. This step repeats for a fixed number of times or till convergence. □

System overview. Algorithm 1 describes the overall process of our solution KAMINO. KAMINO first chooses a schema sequence S based on the schema R, domain $\mathcal{D}$, and DCs Φ (Line 2). Then it finds a suitable parameter set Ψ for the subsequent algorithms to ensure the overall privacy loss is bounded by (ϵ, δ)-DP (Line 3). The algorithms TRAINMODEL$(\cdot)$ and LEARNWEIGHT$(\cdot)$ privately learn the tuple distribution and weights of the DCs from the private true data D^* (Lines 4-5). Last, KAMINO applies a constraint-aware sampling algorithm to generate a synthetic database instance. We first present the key algorithms (Algorithms 4, 2 and 3) when the weights of DCs are given in § 4, and then explain how to learn the DC weights (Algorithm 5) in § 5. Last, privacy analysis and parameter search (Algorithm 6) are explained in § 6.

Our system assumes the inputs are static, since we rely on the database instance to learn the generative process (i.e., Algorithms 2, 3 and 5), and on the DCs to learn the weights and attribute sequence. However, KAMINO can tolerate small input changes as long as the data distribution and DCs are intact. For now, if DC changes resulting in a different sequence, we re-run KAMINO; if the changes significantly shift the distribution, we re-run the generative process. Future work can apply general DP techniques [26] for dynamically growing databases for better utility.

4 KAMINO WITH KNOWN DC WEIGHTS

For simplicity of presentation, in this section, we consider the weights of the constraints are given (e.g., the weights for hard DCs are set infinitely large). We first present our private learning algorithm for the tuple probability and then the database sampling algorithm. Last, we show our choice of schema sequence in KAMINO.

4.1 Private Learning of Tuple Probability

Recall Equ. (2) that, given a schema sequence $S = [A_1, A_2, \ldots, A_k]$, the tuple probability becomes $\Pr[t] = \Pr(t[A_1]) \cdot \prod_{j=2}^{k} \Pr\left(t[A_j] \mid t[A_1, \ldots, A_{j-1}]\right)$. Instead of learning a single distribution over the full domain of a tuple, we learn the probability distribution of the first attribute in the sequence and $(k-1)$ number of conditional probabilities. For the first attribute, we apply Gaussian mechanism (§ 2.4) to learn its distribution. For each of remaining $(k-1)$ condition probabilities, we learn it as a discriminative model. In particular, for each conditional probability $\Pr\left(t[A_j] \mid t[A_1, \ldots, A_{j-1}]\right)$, we train a discriminative sub-model that uses context attributes $(A_1, \ldots, A_{j-1})$ to predict the target attribute A_j. We denote this sub-model by $M_{X,y}$, where $X = S_{:j}$ and $y = S[j]$. We also apply the tuple embedding to privately learn a unified representation with a fixed dimensionality for each attribute in the tuple (§ 2.3). The training of each discriminative sub-model on the samples from the true data is optimized and privatized using DPSGD (§ 2.4).

Algorithm 2 describes how KAMINO privately learns the probability distribution of the first attribute in the sequence S, denoted by $M_{\emptyset,S[1]}$, and the parameters in the $(k-1)$ discriminative sub-models $M_{S_{:j},S[j]}$ for $j \in [2, k]$. It takes as input of the true database instance D^* with domain $\mathcal{D}$, the schema sequence S (to be discussed in § 4.3), as well as learning parameters (number of iterations T, batch size b, learning rate η, and quantizing q bins for numerical attributes) and noise parameters (σ_g and σ_d for Gaussian noise, L_2 norm clip threshold for gradients C). The configuration of these parameters is presented in § 6 to ensure the overall privacy loss of KAMINO is bounded by the given budget (ϵ, δ).

Following the attribute order in S, we start with the first attribute $S[1]$ and apply Gaussian mechanism to the true distribution of $S[1]$ (Line 2-4). If the first attribute has a continuous domain, we partition its domain into q bins. Starting from the second attribute in S, we train the discriminative model. We first load the initial values of the parameters of each sub-model from previous training if they exist (Line 7). Depending on the data type of the target attribute, a cross entropy (for categorical target attribute) or mean squared (for numerical target attribute) loss function on predicting the target attribute value is also set before model training (Line 10).

Each discriminative model $M_{S_{:j},S[j]}$ is learned via backpropagation for T iterations (Line 11-17). At each iteration, we randomly sample a set of training tuples D_e, with sampling probability b/n (i.e., $\mathbb{E}(|D_e|) = b$), and on each of the training tuple, the gradient w.r.t model parameters is computed (Line 13). We clip the L_2 norm of the gradient by the threshold C (Line 14), and add noise to clipped gradient (Line 15) with sensitivity equal to clipping threshold C, before updating the parameters via gradient descent (Line 16). After one discriminative model is trained, we add it to our probabilistic data model M (Line 18). Since we iteratively expand the context attributes as more sub-models are trained, we save the currently trained embeddings of attributes $[X, y]$ (Line 19), and reuse in the initialization of context attributes of the next sub-model (Line 7).

Algorithm 2 Probabilistic data model training

Require: $D^*, \mathcal{D}, S$ ▷ True instance, domain, schema sequence
Require: n, k, η, q ▷ cardinality, dimensions, lr, quantization
Require: σ_g, σ_d ▷ Noise scales in Ψ
Require: C, T, b ▷ L_2 norm clip/#iterations/batch size in Ψ
1: **procedure** TrainModel($D^*, S, \mathcal{D}, \Psi$)
2: $H \leftarrow$ counts of (quantized) values in D^* for 1st attr. $S[1]$
3: Add noise drawn from $\mathcal{N}(0, 2\sigma_g^2)$ to each count in H
4: $M_{\emptyset,S[1]} \leftarrow$ distribution of $S[1]$ based on H, and add it to M
5: Initialize embedding for attribute $S[1]$
6: **for** $j \in [2, k]$ **do**
7: $X = S_{:j}$, load embedding ▷ Context attributes
8: $y = S[j]$, initialize embedding ▷ Target attribute
9: Initialize discriminative model $M_{X,y}$ ▷ [79]
10: $\mathcal{L}(\theta_y, t) \leftarrow$ loss function on imputing target y
11: **for** $e \in [T]$ **do** ▷ For each of iteration
12: $D_e \leftarrow$ random sample on $D^*[X, y]$ with prob b/n
13: For each $t \in D_e$, compute $g_e(t) \leftarrow \nabla_{\theta_y} \mathcal{L}(\theta_y, t)$
14: $\bar{g}_e(t) \leftarrow \max(1, \frac{\|g_e(t)\|_2}{C})$ ▷ Clip gradient
15: $\tilde{g}_e \leftarrow (\sum_{t \in D_e} \bar{g}_e(t) + \mathcal{N}(0, \sigma_d^2 C^2 \mathbb{I}))/b$ ▷ Add noise
16: $\theta_y \leftarrow \theta_y - \eta \times \tilde{g}_e$ ▷ Gradient descent
17: **end for**
18: Add $M_{X,y}$ to M
19: Save embedding and attention weights for $S_{:j+1}$
20: **end for**
21: **return** M
22: **end procedure**

The final output from Algorithm 2 is the probabilistic data model M, which will be used to sample tuple values in § 4.2.

Algorithm 2 consists of $1+(k-1)\times T$ rounds of access to the true database instance D^*. Each access is privatized using the Gaussian mechanism or the DPSGD. By the composibility of differential privacy (§ 2.4), Algorithm 2 satisfies differential privacy. We will analyze the privacy cost in § 6. The time complexity is linear to $n + b(k-1)T$, which is the expected number of tuples that are sampled for training. An optimization for efficiency is to train each $M_{X,y}$ in parallel without reusing previously trained embeddings (Line 7), and we evaluate this trade-off in our full paper [41].

4.2 Constraint-Aware Database Sampling

After we have privately learned the tuple probability, the next step is to sample a database instance D' of size n based on the learned data model M and the given DC weights as summarized in Algorithm 3.

Given a schema sequence S, we first independently sample a value for the first attribute in S of all the n tuples based on its noisy probability distribution represented by $M_{\emptyset,S[1]}$ (Line 2). Depending on $S[1]$'s data type, categorical values are sampled directly; while for numerical values, we first sample a bin, and randomly take a value from the domain represented by the bin.

From the second attribute in S onward, for each attribute A_j and each tuple t_i, we sample a value for $t_i[A_j]$ conditioned on (1) the attributes of t_i that have been assigned a value, i.e., $t_i[S_{:j}] = c$, and (2) the tuples that have been sampled before t_i, i.e. $D'_{:i}[S_{:j+1}]$. For each v from the domain of A_j (or a selected set of values of size

Algorithm 3 Constraint-aware database instance sampling

Require: $S, M, \Phi, \mathcal{D}$▷ Schema sequence, data model, DCs, domain
Require: W, L, N ▷ Weight vector (Alg. 5), sample size, #round
1: **procedure** Synthesize($S, M, \Phi, \mathcal{D}$)
2: $D'[S[1]] \leftarrow$ sample from distribution $M_{\emptyset,S[1]}$
3: **for** $j \in [2, k]$ **do** ▷ Schema sequence S
4: **for** $i \in [1, n]$ **do** ▷ Tuple id sequence
5: $c \leftarrow t_i[S_{:j}]$ ▷ Values for context attributes of t_i
6: $\{p_{v|c} \mid v \in \mathcal{D}(S[j])\} \leftarrow M_{S_{:j}=c,S[j]}$
7: **for** $v \in \mathcal{D}(S[j])$ and $\phi \in \Phi_{S[j]}$ **do**
8: $vio_{\phi,v|D'} \leftarrow$ num. of vio. of ϕ if $t_i[S[j]] = v$
9: **end for**
10: Update $t_i[S[j]] = v$ where v is sampled with $P[v] \propto p_{v|c} \cdot \exp(-\sum_{\phi \in \Phi_{S[j]}} w_\phi \times vio_{\phi,v|D'})$
11: **end for**
12: Resample m random cells $t_r[S[j]]$ or till convergence
13: **end for**
14: **return** D'
15: **end procedure**

d if A_j has a continuous or extremely large domain size), we first extract the conditional probability

$$\Pr(t[A_j] = v \mid t[S_{:j}] = c)$$

from the learned discriminative sub-model $M_{S_{:j},S[j]}$, and denote it by $p_{v|c}$ (Line 6). If the target attribute A_j has a discrete domain, the conditional probability $p_{v|c}$ takes the probability that $M_{S_{:j},S[j]}$ predicts the target attribute $A_j = v$ given the context attributes $S_{:j} = c$. If the target attribute A_j has a continuous domain, the discriminative model is based on regression model and outputs a Gaussian distribution mean μ and std σ given the context attributes $S_{:j} = c$. We sample d number of candidates from this distribution and assign each candidate v with a probability $p_{v|c} \propto \{\frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2}(\frac{v-\mu}{\sigma})^2)\}$. The other values in the domain are assigned with probability 0. We denote the candidate set by $\mathcal{D}(S[j])$.

Next, we compute the number of DC violations $vio_{\phi,v|D'}$ if we assign $t_i[A_j] = v$:

$$|V(\phi, t_i[S_{:j}] = c \wedge t_i[A_j] = v \mid D'_{:i}[S_{:j+1}])|$$

for each DC violation $\phi \in \Phi_{A_j}$ (Line 8). Last, we sample a value v based on the combined probability

$$P[v] \propto p_{v|c} \cdot \exp(-\sum_{\phi \in \Phi_{A_j}} w_\phi \times vio_{\phi,v|D'})$$

and update the jth attribute of t_i (Line 10). The final output is a synthetic database instance D' of size n with the same schema as the true database instance D^*.

Without the constraint-aware sampling (Line 7-9), the sampling process results in a set of i.i.d. tuple samples. This resulted instance can fail to preserve even simple constraints such as FDs (e.g., ϕ_1) or single-tuple DCs (e.g., ϕ_3), because not all the domain values appear in the true data D^*. Such values can be sampled due to noisy distribution and hence lead to DC violations. By adjusting the sampling probability based on the violations caused by the new cell value of a tuple (Line 10), we can control the additional number of violations due to the noisy distribution learned.

Algorithm 4 Constraint-aware attribute sequencing

Require: $R, \mathcal{D}, \Phi$ ▷ Input schema, domain, and DCs
1: **procedure** SEQUENCING($R, \mathcal{D}, \Phi$)
2: $\Sigma \leftarrow$ FDs from Φ sorted by increasing domain size of LHS
3: Initialize $S \leftarrow []$
4: **for all** $X \rightarrow Y \in \Sigma$ **do**
5: Sort attributes X by domain size
6: For all $A \in [X, Y]$, append A to S if $A \notin S$
7: **end for**
8: Append attributes in $(R - S)$ to S in an order of increasing domain size, and **return** S
9: **end procedure**

General MCMC sampling requires re-sampling of the entire full D' with all attributes, and hence at least $k-1$ more conditional distributions need to be learned. However, in the private setting with a fixed privacy budget, learning more distributions will compromise the accuracy of each learned distribution. Therefore, KAMINO uses a constrained MCMC based on the same set of conditional distributions. As we loop over each attribute (Line 3-13), it re-samples random cell values for this attribute, conditioned on all other sampled values (Line 12).

The time complexity of checking one DC's violations for all n values is $O(dn)$ (for an unary DC) or $O(dn^2)$ (for a binary DC). This can be optimized by exploiting the property of hard functional dependencies, and we evaluate this optimization in our full paper [41]. In addition, when $m > 0$ for MCMC, the sampling algorithm has an additional cost of $O(mkd + |\Phi|dnm)$. The overall complexity of constraint-aware sampling is $O(nkd + |\Phi|dn^2 + mkd + |\Phi|dnm)$.

4.3 Constraint-Aware Sequencing

Given a fixed privacy budget, the goal is to identify a good schema sequence, where the set of attributes that can well discriminate attribute A_j should appear before A_j in the sequence. Unlike prior work [25, 82] that spend part of the privacy budget in learning a good sequence, we make use of the input DCs Φ and the domain $\mathcal{D}$. This heuristic approach incurs no privacy cost since the true database instance D^* is not queried.

Specifically, we propose a rule-based, instance-independent method to ensure that for an FD $X \rightarrow Y$ in Φ, we have X ahead of Y in S (unless $Y \rightarrow X$ too). Algorithm 4 describes the process of finding a schema sequence S. For the list of FDs $\Sigma = [X_1 \rightarrow Y_1, \ldots, X_m \rightarrow Y_m]$, we sort the list Σ by the minimal domain size of an attribute from X (i.e., $\exists A^1 \in X_1, \forall A^2 \in X_2, |\mathcal{D}(A^1)| \leq |\mathcal{D}(A^2)|$) (Line 2). For each FD, we greedily add its left hand side and right hand side attributes into the final schema sequence S (Line 4-7). For the rest of attributes that do not participate in FDs, we order them by ascending domain size and append to S (Line 8). The complexity is $O(k|\Sigma| + \log k)$, consisting costs of sorting FDs and attributes.

Our sequencing algorithm relies on the given FDs as a subset of DCs. In cases that Φ does not include any FDs (i.e., $\Sigma = \emptyset$), Algorithm 4 returns a sequence based on the domain size. Following this sequence, each discriminative sub-model (§ 4.1) will have the smallest possible domain size for its context attributes (cross-product of all context attributes' domain sizes), and hence each sub-model can be more accurately learned. For example, consider $[A_1, A_2, A_3]$ with domain sizes 2, 3, 5, respectively. The overall context attribute domain size is 8 (=2+6), instead of 20 on the reversed sequence.

Algorithm 5 Learning DC weights

Require: D^*, Φ, S ▷ True instance, DCs, schema sequence
Require: σ_w, T_w, L_w ▷ Noise scale/#iteration/sample size in Ψ
Require: b_w, S_w ▷ Batch size in Ψ, sensitivity
1: **procedure** LEARNWEIGHT(D^*, Φ, S, M, Ψ)
2: Initialize weight vector W of length $|\Phi|$ if unknown
3: Take a random sample $\hat{D}$ from D^* with a probability L_w/n
4: Drop tuples from the sample if $|\hat{D}| > L_w$
5: Compute violation matrix V of size $(|\hat{D}| \times |\Phi|)$ from $\hat{D}$
6: Add noise drawn from $\mathcal{N}(0, S_w^2\sigma_w^2)$ to each value in V
7: Set negative values in V to zero
8: **for** $A_j \in S$ and $e \in [T_w]$ **do**
9: $ids \leftarrow$ sample b ids from $[1, L_w]$ with prob b_w/L_w
10: **for** each $i \in ids$ **do**
11: $O \leftarrow \exp(-\sum_{\phi_l \in \Phi_{A_j}} W[l] \cdot V[i][l])$
12: Update W via back propagation by max O
13: **end for**
14: **end for**
15: **return** W
16: **end procedure**

Optimizations for extreme domain sizes. For attributes with small domain size, we can group adjacent attributes in the schema sequence into one hyper attribute, and learn one discriminative sub-model instead of multiple sub-models. As a result, less privacy budget will be consumed. For example, applying Algorithm 4 on the BR2000 dataset [83] with 38k tuples resulted in a schema sequence starting with 7 binary attributes. In this case, we can create a hyper attribute of domain size 2^7 to replace the group of the binary attributes. After the synthetic hyper attribute value is generated, we can un-group it to individual attributes and check violations if any. On the other end, the distribution of attributes with very large domain size may not be learned well, due to insufficient amount of training data. For example, the Tax dataset [24] with 30k tuples has one *zip* attribute with domain size of 18k. The training sample of size $b \times T$ in Algorithm 2 may not cover all values in the domain, and hence learned distribution can have large variance. In this case, we can apply Gaussian mechanism to its true distribution, and sample independently without relying on the context attributes.

5 LEARNING DC WEIGHTS

KAMINO so far assumes the weights of DCs W are known. For example, the weights for hard DCs (no violations in the true data) are set to be infinitely large. However, for soft DCs, the weights are usually unknown and need to be estimated. We follow the intuition that if a DC is observed with many violations in the training data, then its weight will be set small. Otherwise, if there is no violation, then its weight will be set large. Based on this intuition, we design Algorithm 5 to first privately learn the number of violations to each DC and then estimate the weights as a post-processing step.

We transform the given data instance D into a violation matrix V of size $|D| \times |\Phi|$, where each value $V[i][l]$ represents the number

Algorithm 6 Searching DP parameters

Require: $\epsilon, \delta, \mathcal{D}, S$ ▷ Privacy budget, domain, schema sequence

1: **procedure** SearchDParas($\epsilon, \delta, \mathcal{D}, S$)
2: Initialize parameter configuration Ψ with a default setting $(\sigma_g.min, \sigma_d.min, \sigma_w.min, b.max, T.max, |S|, L_w.max, \ldots)$
3: $\Psi.b_w \leftarrow 1$ if DC weights are unknown
4: **while** $\epsilon_\Psi(\delta) > \epsilon$ **do** ▷ Cost of Kamino [41]
5: If $\Psi.T > T_{min}$, then decrease $\Psi.T$
6: If $\Psi.\sigma_d < \sigma_{d_{max}}$, then increase $\Psi.\sigma_d$
7: If $\Psi.\sigma_g < \sigma_{g_{max}}$, then increase $\Psi.\sigma_g$
8: If $\Psi.b > b_{min}$, then decrease $\Psi.b$
9: ...
10: **end while**
11: **return** Ψ
12: **end procedure**

of violations to the lth DC in Φ caused by tuple t_i with respect to all other tuples in D, i.e., $V(\phi_l, t_i \mid D - \{t_i\})$. Based on the transformed data, the objective is to maximize the exponential part represented in Eqn. (1). However, the violation matrix based on the full true instance is highly sensitive to the change of one tuple. For binary DCs that involve two tuples, changing one tuple can incur up to $O(n)$ additional number of violations.

To bound the sensitivity of the violation matrix, we sample a small set of tuples $\hat{D}$ of size L_w as the training example (Line 4). Each tuple from the true instance D^* is independently sampled with probability L_w/n (i.e., $\mathbb{E}(|\hat{D}|) = L_w$). If the resulted sample has a size greater than L_w, we randomly drop tuples to crop the size to L_w. This allows us to bound the sensitivity of the violation matrix, and also reduces the time complexity from $O(|\Phi|n^2)$ to $O(|\Phi|L_w^2)$. The sensitivity analysis on S_w can be found in our full paper [41].

Hence, we apply Gaussian mechanism to perturb the violation matrix V over the samples and post-process all the negative noisy counts to zeros (Lines 5-7). Then we loop over each attribute $A_j \in S$ for T_w iterations (Line 8). For each A_j, we sample b rows from the noisy V to update weights W for the set of active DCs related to A_j (Lines 8-14). We will analyze the privacy cost in § 6. The time complexity of this post-processing step is $O(|\Phi|bT_w)$ in terms of the number of tuples that are used for learning.

6 PRIVACY ANALYSIS

Kamino involves at most three processes that require access to the true database instance:

M_1: Learning the distribution of the first attribute in the schema sequence (Algorithm 2 Line 2-4);

M_2: Training $k-1$ discriminative models (Algorithm 2 Line 6-20);

M_3: Learning the DC weights if unknown (Algorithm 5).

Each process has been privatized using the Gaussian mechanism or DPSGD. The other steps (Algorithm 3 and Algorithm 4) not accessing the true database do not incur privacy loss. Hence, we can show Kamino achieves DP by simple sequential composition [28] and post-processing property [29] of DP. In our full paper [41], we give a tighter privacy bound using *Rényi DP* (RDP) [60] and prove the privacy of Kamino.

Table 1: Description of the datasets.

Dataset	n	k	Domain size	Hard DCs	DC IDs[2]
Adult	32,561	15	$\approx 2^{52}$	Yes	$\{\phi^a_{1-2}\}$
BR200	38,000	14	$\approx 2^{16}$	No	$\{\phi^b_{1-3}\}$
Tax	30,000	12	$\approx 2^{71}$	Yes	$\{\phi^t_{1-6}\}$
TPC-H	20,000	9	$\approx 2^{42}$	Yes	$\{\phi^h_{1-4}\}$

In practice, the overall privacy budget (ϵ, δ) is specified as an input to Kamino, and one needs to judiciously set the privacy parameters in Ψ. Setting these parameters is non-trivial as they are volatile to input datasets. To automatically assign parameters, Kamino provides a parameter search algorithm, summarized in Algorithm 6. It takes the privacy budget (ϵ, δ) and outputs a set of parameters Ψ that ensures that the overall privacy cost does not exceed (ϵ, δ). It starts with a default setting based on prior experimental heuristics [14, 76] and the domain information $\mathcal{D}$. The noise parameters including $(\sigma_g, \sigma_d, \sigma_w, b, T, L_w)$ are boldly set to give the best possible accuracy (Line 2). If this privacy cost of this configuration is higher than ϵ (Line 4), then we use a priority order to decide which parameter to tune (Lines 5-9). This process is repeated till the privacy loss is capped at our total budget. The time complexity is linear to the size of parameter space.

The parameter settings can be found in our full paper [41].

7 EVALUATION

In this section, we evaluate the synthetic data generated by Kamino with three utility metrics: (i) consistency with DC constraints in the true data; (ii) usefulness in training classification models; and (iii) accuracy in answering α-way marginal queries. We show that:

- Kamino preserves data consistency, while state-of-the-art methods fail to preserve most DCs. Kamino is practically efficient.
- While Kamino is not designed for particular tasks, it can achieve comparable and even better quality in the learning and query task, compared to methods that are designed for these tasks.
- The constraint-aware sampling and sequencing are effective to keep data consistency.
- Kamino scales linearly with the number of DCs.

7.1 Evaluation Setup

Datasets. We choose 4 different datasets with mixed data types and DCs, listed in Table 1. First, the Adult dataset [27] consists of 15 census attributes and 2 hard DCs. Second, the BR2000 dataset [83] has a smaller domain size than the Adult dataset, but it has 3 soft DCs with unknown weights. The third dataset, Tax [24], has a very large domain size, e.g., *zip* ($\approx 2^{15}$) and *city* ($\approx 2^{14}$) and 6 hard DCs. Last, TPC-H [7], a synthethic dataset that joins three tables (Orders, Customer and Nation) and removes unique attributes such as *orderkey* and *comment*. The final table consists of 20,000 orders with 9 numerical and categorical attributes. The set of hard DCs are obtained by the foreign key and primary key constraints.

Baselines. Four state-of-the-arts to allow the synthesis of relational data with DP guarantees are considered: 1) PrivBayes [83], a

[2]The sets of DCs are listed in the full paper [41].

statistical method based on Bayesian network; 2) PATE-GAN [50], a GAN-based method that trains a data generator using the PATE's student-teacher model [63]; 3) DP-VAE [20], which samples from the latent space of a privately trained auto-encoder [51]; and 4) The winning solution of the NIST challenge [61] (labeled as NIST), which applies probabilistic inference [57] over marginals.

PATE-GAN and DP-VAE require the input dataset to be encoded into numeric vectors, and we apply the best encoding scheme empirically [35]. Additionally, PATE-GAN requires one labeled attribute to train a set of conditional generators, where each generator produces synthetic data conditioning on one value in the domain of the labeled attribute. We choose the attribute with smallest domain size from each dataset as the labeled attribute, and generate the same number of tuples as in the true data, although it reveals the true histogram of the labeled attribute and favors answering marginal queries. Finally, NIST requires a set of marginals as input for inference. We use marginals over every single attribute, and over 10 randomly chosen attribute pairs.

Evaluation Metrics. We evaluate a synthetic database instance D' of the same size as the true data D^* using three metrics.

Metric I: DC Violations. Since all known DCs are binary, we measure the percentage of tuple pairs that violate DCs in an instance D of size n, i.e., $100 \cdot |V(\phi, D)|/\binom{n}{2}$.

Metric II: Model training. We consider 9 classification models (LogisticRegression, AdaBoost, GradientBoost, XGBoost, RandomForest, BernoulliNB, DecisionTree, Bagging, and MLP). On every single attribute of a dataset, we train all models to classify one binary label (e.g., income is more than 50k or not, age is senior or not, occupation is government job or not) using all other attributes as features. The quality of the learning task on one attribute is represented by the average of all models. Accuracy and F1 are reported for learning quality. Each model is trained using 70% of the synthetic database instance, and evaluate the accuracy and F1 using the same 30% of the true database instance. We also show the results of training and testing on the true dataset labeled as Truth.

Metric III: α-way marginals. For each attribute combination $\mathbb{A}$, we compute the α-way marginal, $h : \mathcal{D} \to \mathbb{R}^{|\mathcal{D}(\mathbb{A})|}$ on the synthetic data D' and true data D^*, respectively, and then report the total variation distance [74] as $\max_{a \in \mathcal{D}(\mathbb{A})} |h(D')[a] - h(D^*)[a]|$.

Implementation details. Kamino was implemented in Python 3.6 and tested with $m = 0$ by default. For the discriminative sub-models, we integrated the code [3] from AimNet in the HoloClean system. For the baselines, we reused the code [2, 4, 5] from their authors with all default parameters. All the 9 models in the learning task were implemented using standard libraries [6, 22] and trained with default parameters, except that we set `random_state` = 0 whenever possible, for the purpose of reproducibility. We report the mean and standard deviation of 3 runs for each test. All experiments were conducted on a machine with 12 cores and 64GB RAM. The code, data and evaluation metrics are open sourced on GitHub: https://github.com/cgebest/kamino.

7.2 End-to-End Evaluation

We compare Kamino with all four baselines at a fixed privacy budget ($\epsilon = 1, \delta = 10^{-6}$).

Table 2: Percentage of tuple pairs that violate DCs. Kamino has the closet DC violations as the truth, while none of the baselines are able to preserves most of the DCs.

DC	Truth	PrivBayes	DP-VAE	PATE-GAN	NIST	Kamino
ϕ_1^a	0.0	11.3±0.3	32.0±0.2	20.3±0.0	0.0±0.0	**0.0±0.0**
ϕ_2^a	0.0	1.4±0.6	13.2±0.1	24.8±0.1	0.0±0.0	**0.0±0.0**
ϕ_1^b	0.4	1.6±0.0	0.0±0.0	**0.4±0.0**	0.0±0.0	0.6±0.0
ϕ_2^b	0.9	2.6±0.2	15.6±0.2	0.2±0.0	28.1±6.8	**0.6±0.0**
ϕ_3^b	0.5	1.4±0.1	0.0±0.0	0.1±0.0	0.0±0.0	**0.3±0.2**
ϕ_1^t	0.0	0.0±0.0	0.0±0.0	0.0±0.0	7.4±1.3	**0.0±0.0**
ϕ_2^t	0.0	0.8±0.0	0.0±0.0	0.8±0.0	0.4±0.0	**0.0±0.0**
ϕ_3^t	0.0	0.0±0.0	0.0±0.0	0.0±0.0	8.0±1.7	**0.0±0.0**
ϕ_4^t	0.0	0.4±0.0	98.9±0.0	2.1±0.0	0.0±0.0	**0.0±0.0**
ϕ_5^t	0.0	0.5±0.0	99.0±0.0	4.0±0.0	0.0±0.0	**0.0±0.0**
ϕ_6^t	0.0	0.4±0.0	24.5±0.1	0.9±0.0	0.0±0.0	**0.0±0.0**
ϕ_1^h	0.0	0.2±0.0	16.7±0.2	5.1±0.1	64.0±45.2	**0.0±0.0**
ϕ_2^h	0.0	0.2±0.0	15.7±0.1	4.4±0.1	53.4±37.7	**0.0±0.0**
ϕ_3^h	0.0	0.2±0.0	15.3±0.2	5.1±0.1	64.0±45.2	**0.0±0.0**
ϕ_4^h	0.0	0.6±0.0	30.1±0.1	1.2±0.0	3.2±0.0	**0.0±0.0**

7.2.1 Experiment 1: DC Violations. We show that synthetic data generated by Kamino has a similar number of DC violations as the true database instance. Table 2 lists the percentage of tuple pairs that violate each of the given DC. On the Adult, Tax and TPC-H datasets, Kamino incurs zero violations, which is consistent to the observations in the true database instances. On the BR2000 dataset, the overall numbers of DC violations on the synthetic instance output by Kamino are the closest to those on the truth among all approaches. The baselines fail to preserve most of the DCs. For instance, the hard DC ϕ_1^a on the Adult dataset has about 11.3%, 32%, and 20.3% violations in the synthetic data generated by PrivBayes, DP-AVE and PATE-GAN, respectively. Although NIST does not have violations like Kamino, it it because NIST filled the entire *edu_num* column with the same value. For another instance, all the hard DCs induced by the foreign key and primary key constraints in the TPC-H dataset, are preserved only in Kamino.

7.2.2 Experiment 2: Model Training. Figure 3 shows the accuracy and F1 on classifying all attributes. Each data point in Figure 3 represents an average of 9 classification models for classifying one target attribute, and we use the box plot to show classification quality on all attributes for each of the dataset. As Figure 3 shows, Kamino achieves the best overall accuracy and F1 on most datasets: the mean of all attributes in Kamino is the closest to the truth, and other quartiles are the best for majority of the tests comparing to the baseline systems. For instance, on Adult, training and testing on the true database instance gives average accuracy of 0.88. The models on the synthetic data by Kamino is 0.82, which outperforms PATE-GAN (0.77), PrivBayes (0.68), NIST (0.66), and DP-VAE (0.54).

7.2.3 Experiment 3: α-way Marginals. Figure 4 shows the total variation distance for all attributes or attribute combinations on each of the dataset. Each data point represents a total variation distance of the distributions between the true database instance and the synthetic database instance, for a certain attribute (1-way) or an attribute set (2-way). As it shows, Kamino has the smallest or close to the smallest variation distances. Taking the first 1-way marginal on the Adult dataset as an example, Kamino has a mean

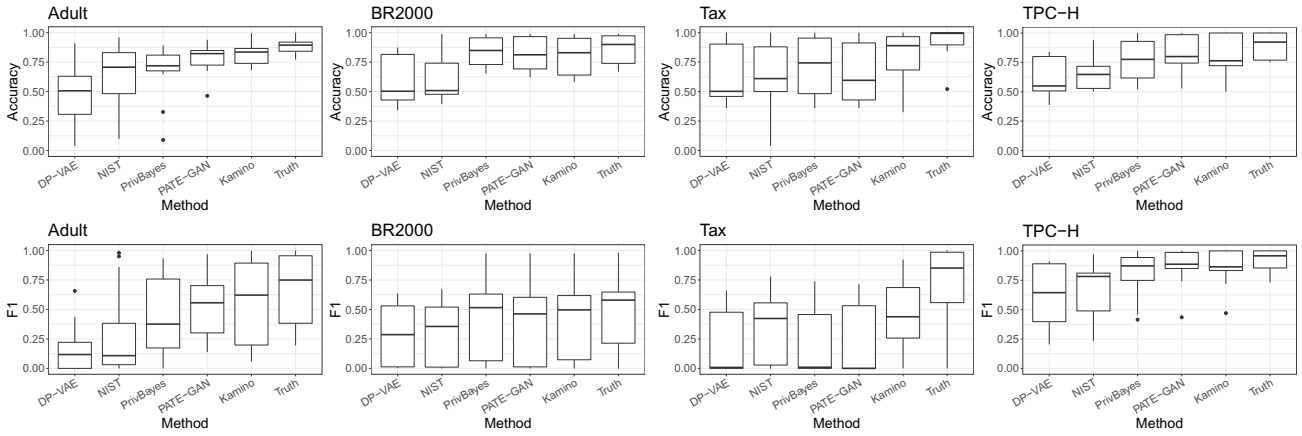


Figure 3: Accuracy and F1 of evaluating classification models, which are tested on the true dataset and trained on synthetic data by different methods. Each point represents an averaged classification quality (accuracy or F1) over 9 models for one target attribute using all other attributes as features. Each box represents a set of classifications, one for each attribute in the schema. KAMINO achieves the overall best accuracy and F1 scores on most datasets.

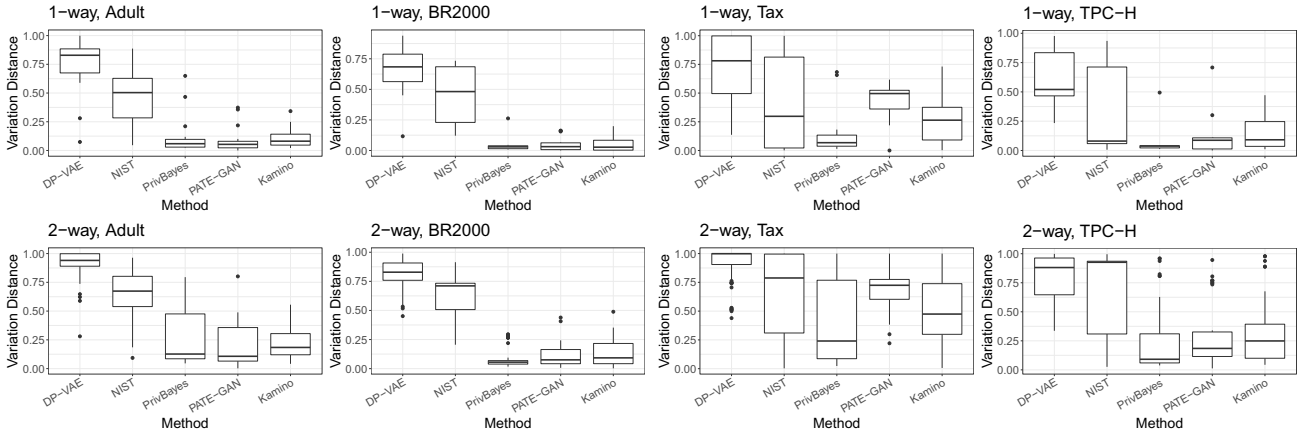


Figure 4: Total variation distance on α-way marginals, where $\alpha = [1, 2]$. Each point represents a total variance distance for one attribute set, and each box represents total variance distance for all attribute sets. It shows that KAMINO can achieve overall the best (Adult) or close to the best (BR2000, Tax and TPC-H) variation distance.

of 0.11, which is second to the smallest mean of PATE-GAN (0.09), and a maximal distance of 0.34, which is the smallest comparing to PATE-GAN (0.37), PrivBayes (0.65), NIST (0.89), and DP-VAE (1.0).

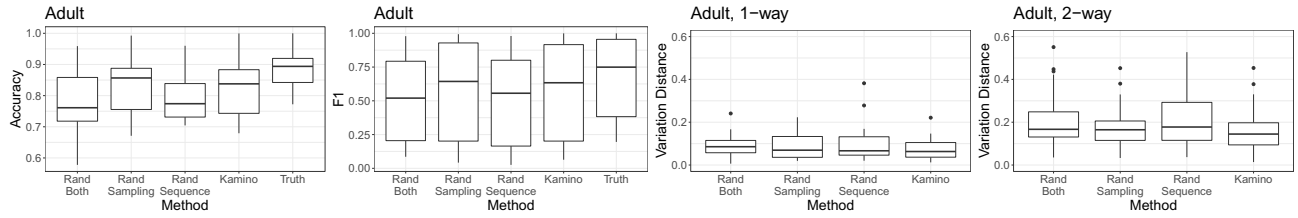
7.2.4 Experiment 4: Execution time. Since KAMINO explicitly checks DC violations during sampling, it is expected to take longer running time than baseline methods that generate i.i.d samples. In our evaluation, NIST and PrivBayes were the most efficient on all datasets, and took at most 217±13 and 1,367±561 seconds, respectively. Because of training deep models on encoded data, running time of DP-VAE and PATE-GAN on all datasets fell into the range of 20 minutes to 13 hours. For KAMINO, the running time on all datasets were in 5-16 hours, which is still practically efficient.

Figure 7 profiles KAMINO's execution time of each process (sequencing, model training, computing violation matrix and learn DC weights for soft DCs, and sampling). As Figure 7 shows, performance of KAMINO is dominated by training and sampling, which together take more than 99% of the total time. MCMC re-sampling further increases the sampling time, but it leads to better task qualities. We show the detailed evaluation on MCMC re-sampling in our full paper [41].

In addition, the full paper also include evaluations of optimization techniques, which can speed up model training on Adult by 3.5×, and allow KAMINO to complete in 10 hours for a TPC-H dataset that scaled up to 1 million rows.

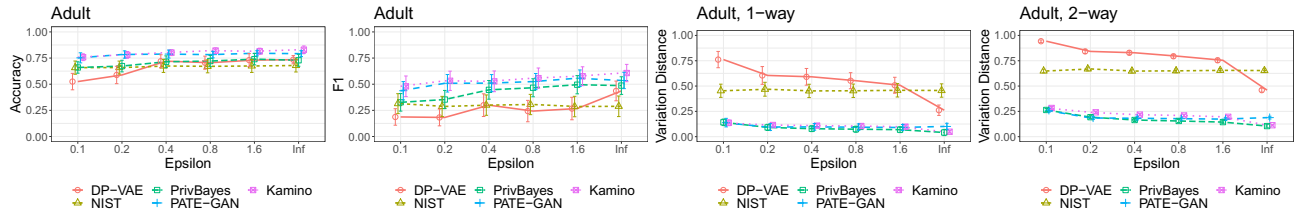
7.3 Component Evaluation

7.3.1 Experiment 5: Effectiveness of constraint-aware components. Recall that our approach takes DCs into account when it samples synthetic values (§ 4.2) and generate the schema sequence (§ 4.3). In this experiment, we compare KAMINO with three sub-optimal KAMINO that do not have the constraint-aware components:



(a) Accuracy (b) F1 (c) 1-way marginal (d) 2-way marginal

Figure 5: Accuracy and F1 of model training on KAMINO, and sub-optimal KAMINO without constraint-aware sampling, sequencing, and neither, using the Adult dataset as the example. It shows the the KAMINO with constraint-aware components can achieve the best quality in both the learning task and in the query task.



(a) Accuracy (b) F1 (c) 1-way marginal (d) 2-way marginal

Figure 6: Task quality of the KAMINO and baselines by varying privacy budget $(\epsilon, 10^{-6})$.

Table 3: Percentage of DC violations using KAMINO, and sub-optimal KAMINO w/o constraint-aware components.

DC	Truth	KAMINO	RandSequence	RandSampling	RandBoth
ϕ_1^a	0	0.0±0.0	0.0±0.0	0.4±0.0	9.1±8.5
ϕ_2^a	0	0.0±0.0	0.0±0.0	36.8±0.3	26.1±11.0

- Replace constraint-aware sampling (Algorithm 3) in KAMINO with sampling tuples independently, labeled as "RandSampling";
- Replace constraint-aware sequencing (Algorithm 4) by a random sequence, labeled as "RandSequence";
- Replace both components above, labeled as "RandBoth".

Table 3 compares DC violations of the synthetic data generated by KAMINO and by sub-optimal KAMINO without constraint-aware components. First, we see that without constraint-aware sampling component (Algorithm 3), the synthetic data generated by RandSampling and RandBoth have more violations than the other two methods. Second, the constraint-aware sequencing component (Algorithm 4) is also important. Take ϕ_1^a : $edu \rightarrow edu\_num$ as an example, RandBoth (without the constraint-aware sequencing) results in a higher number of DC violations than RandSampling. This is because that *edu* is not necessarily placed before *edu_num* in a random schema sequence, and the noisy model cannot preserve the correlation between these two attributes. Similar, without constraint-aware components, quality downgrades in both learning and query task shown in Figure 5.

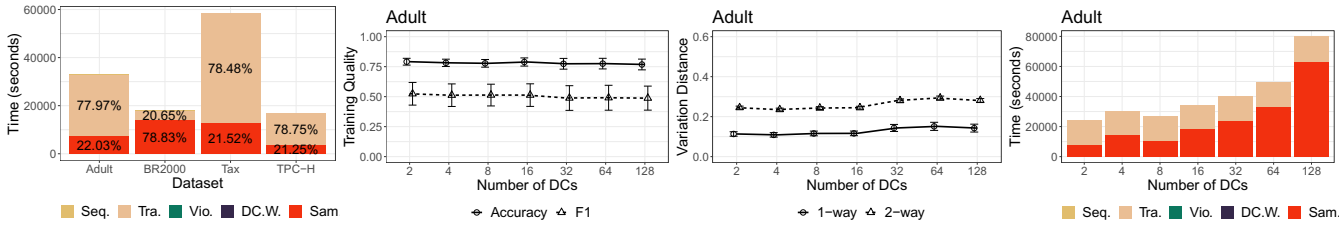
We omit the presentation of non-private runs for similar observations. We believe that the constraint-aware components can also be incorporated into the baseline systems, but we skip the comparison because it requires significant re-design of the baseline systems.

7.3.2 Experiment 6: KAMINO vs Accept-Reject Sampling. KAMINO's constraint-aware sampling (Algorithm 3) explicitly constructs the target distribution and directly samples from it for filling a cell (Line 10). Another sampling method is the accept-reject (AR) sampling [62], which samples one value at a time, and accepts this value probabilistically based on its violations. For soft DCs, AR-sampling can be an alternative, but it does not work well for hard DCs.

We first evaluate KAMINO using AR-sampling on the Adult dataset with hard DCs. AR-sampling does not work well when hard DCs are present. If a sampled value incurs any violations, then its accept ratio (i.e., $\exp(-\sum_{\phi \in \Phi_{A_j}} w_\phi \times vio_{\phi,v|D'})$, where v is the sampled value of attribute A_j) diminishes to 0, since $w_\phi = \infty$. As a result, AR-sampling needs re-sampling multiple times until a value can be accepted, depending on the other cells that have been filled with sampled values. For efficiency purpose, we allow at most 300 samples per cell: if no values can be accepted, we take the last sampled value and as a result, violations can occur. KAMINO with AR-sampling does produce violations for the two DCs ϕ_a^1 (0.4±0.0) and ϕ_a^2 (37.2±0.0). The execution time of KAMINO with AR-sampling takes 7.5 hours, which is 1.9× longer.

On the BR2000 dataset with soft DCs, KAMINO with AR-sampling completes in 1.26 hours (0.24 hour for the AR-sampling step) on average, which is faster than the constraint-aware sampling (3.9 hours). AR-sampling converges faster due to its relatively high accept ratio. For DC violations and task qualities, we observe that KAMINO with AR-sampling performs similarly with KAMINO.

7.3.3 Experiment 7: Varying Privacy Budget. We show the impact of the privacy budget in the task qualities using the Adult dataset as the example. Figure 6 compares the data usefulness by varying the privacy budget parameter (ϵ, δ) at different $\epsilon = [0.1, 0.2, 0.4, 0.8, 1.6]$ with a constant $\delta = 10^{-6}$. $\epsilon = \infty$ refers to non-private KAMINO and



(a) Model Training (b) Marginal Distance (c) Time Complexity

Figure 7: Time profiling of end-to-end runs on all datasets.

Figure 8: Task quality and execution time by varying the number of DCs.

baselines. First of all, increasing the privacy budget leads to overall better quality in both the learning and the query tasks. Consistent with the observations in Figures 3-4, Kamino always achieves the best in training quality (Figures 6a-6b) and close to best marginal distances (Figures 6c-6d) at different privacy budgets. The averaged model accuracy over all attributes on Kamino is 0.8 at privacy budget ($\epsilon = 0.2, \delta = 10^{-6}$), which outperforms DP-VAE (0.54), NIST(0.66), PrivBayes (0.68) and PATE-GAN (0.77) at 5× larger $\epsilon = 1$.

7.3.4 Experiment 8: Scalability of DCs. In this experiment, we vary the number of DCs from the input to Kamino. Due to the lack of large numbers of ground DCs, we generate the input DCs by discovering approximate DCs [64] to simulate the knowledge from the domain expert.

Figure 8 shows the task quality and time profiling as increasing the number of soft DCs from 2 to 128, under the fixed privacy budget ($\epsilon = 1, \delta = 10^{-6}$) on the Adult dataset. Since the DC weights are noisy and approximately learned using a subset of data (Algorithm 5), increasing the number of DCs implies more noisy adjustment for the sampling probabilities (Algorithm 3). As a result, task quality is expected to decrease given a finite privacy budget. Figure 8a and Figure 8b show that as the number of DCs increases to 128, task quality only degrades by 0.04.

As the number of DCs increases, more time is required to compute the violation matrix, learn DC weights, and to sample. In particular, for Kamino's constraint-aware sampling process (Algorithm 3), introducing more DCs will linearly increases the time to check DC violations for each of the DCs. Since the total execution time is dominated by the sampling process, the total execution time of Kamino scales linearly with the number of DCs. Figure 8c shows that when the number of DC increases from 2 to 128, the total execution time increases only by 3×.

8 RELATED WORK

There has been extensive literature on releasing differentially private synthetic data [17, 36, 61, 84]. These approaches can be categorized into two classes: 1) statistical approaches, which focus on synthesizing low-dimensional projections; and 2) deep learning approaches, which train a deep generative model to sample tuples. Both classes assume tuples are i.i.d, and hence cannot preserve the structure of the data. Our approach is a combination of both, and more importantly, our method differentiates prior work in that we explicitly consider the denial constraints [48] enforced among tuples, rather than simply assuming tuple independence.

Statistical approaches for generating synthetic data usually estimate low-dimensional marginal distributions [66, 80], due to the hardness of privatizing high-dimensional data with differential privacy guarantee [16, 31, 39, 75]. These low-dimensional distributions can be used to estimate the high-dimensional tuple distribution, based on the assumption of conditional independence among attributes, which can be modeled using probabilistic graphical models [52], such as using the Bayesian network [56, 65, 83] or undirected graphs [21, 57]. Under this model, only correlations among dependent attributes are likely to be captured, but correlations that widely exist among conditional independent attributes and tuples are not captured in prior work.

Deep learning models have been shown widely used in synthesizing unstructured data, such as images [70], videos [19] and natural languages [44]. Different from unstructured data, structured data is defined using relational schema and hence, stucture correlations naturally exist. Naïvly applying deep learning models such as GAN [42] and auto-encoder [51] on structured data faces at least two challenges. First, those models usually take numeric vectors as input, and popular encoding schemes such as one-hot encoding or ordinal encoding do not work well on structured data [35], Second, similar to statistical approaches, methods based on deep models (e.g. [38, 50, 73, 81]) suffer from missing structure correlations.

In general, generating differentially private synthetic data is hard, due to the tradeoff between accuracy and privacy [16, 31, 39, 75]. On the other hand, an efficient private data generation algorithm fails to offer the same level of accuracy guarantees to all the queries. Existing practical methods (e.g., [12, 20, 21, 50, 83]) therefore choose to privately learn only a subset of correlations to model the true data. However, the structure of the data is not explicitly captured by these methods and thus are poorly preserved in the outputs.

9 CONCLUSION

In this work, we are motivated to design a synthetic data generator that can preserve both the structure of the data, and the privacy of individual data records. We present Kamino, an end-to-end data synthesis system for constraint-aware differentially private data synthesis. Kamino takes as input a database instance, along with its schema (including denial constraints), and produces a synthetic database instance. Experimental results show that Kamino can preserve the structure of the data, while generating useful synthetic data for applications of training classification models and answering marginal queries, comparing to the state-of-the-art methods.

REFERENCES

[1] 2016-04-27. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *OJ* (2016-04-27).

[2] Online. Code for the winning solution in Differential Privacy Synthetic Data Challenge. https://github.com/usnistgov/PrivacyEngCollabSpace/tree/master/tools/de-identification/Differential-Privacy-Synthetic-Data-Challenge-Algorithms/rmckenna

[3] Online. HoloClean code. https://github.com/HoloClean/holoclean/

[4] Online. PATE-GAN code. https://bitbucket.org/mvdschaar/mlforhealthlabpub/src/master/alg/pategan/

[5] Online. PrivBayes code. https://sourceforge.net/projects/privbayes/

[6] Version 0.23.2. scikit-learn, Machine Learning in Python. https://scikit-learn.org/

[7] Version 2.18.0. The TPC Benchmark H (TPC-H). http://www.tpc.org/tpch/

[8] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *CCS*. ACM, 308–318.

[9] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In *KDD*. 2867.

[10] Brooke Auxier, Lee Rainie, Monica Anderson, Andrew Perrin, Madhu Kumar, and Erica Turner. 2019. Americans and Privacy - Concerned Confused and Feeling Lack of Control Over Their Personal Information. *Pew Research Center* (2019).

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[12] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. 2007. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*. 273–282.

[13] Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. 2014. Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. In *FOCS*. 464–473.

[14] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* 13 (2012), 281–305.

[15] Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. 2017. Efficient Denial Constraint Discovery with Hydra. *PVLDB* 11, 3 (2017), 311–323.

[16] Avrim Blum, Katrina Ligett, and Aaron Roth. 2008. A learning theory approach to non-interactive database privacy. In *STOC*. ACM, 609–618.

[17] Claire McKay Bowen and Fang Liu. 2020. Comparative Study of Differentially Private Data Synthesis Methods. *Statist. Sci.* 35, 2 (May 2020), 280–307. https://doi.org/10.1214/19-sts742

[18] U.S. Census Bureau. Accessed on 2020-11-30. LEHD Origin-Destination Employment Statistics (2002-2017). https://onthemap.ces.census.gov/

[19] R. Chawla. 2019. Deepfakes : How a pervert shook the world. *International Journal for Advance Research and Development* 4 (2019), 4–8.

[20] Qingrong Chen, Chong Xiang, Minhui Xue, Bo Li, Nikita Borisov, Dali Kaafar, and Haojin Zhu. 2018. Differentially Private Data Generative Models. *CoRR* abs/1812.02274 (2018).

[21] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. 2015. Differentially Private High-Dimensional Data Publication via Sampling-Based Inference. In *SIGKDD*. 129–138.

[22] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *SIGKDD*. ACM, 785–794.

[23] David Maxwell Chickering. 1995. Learning Bayesian Networks is NP-Complete. In *AISTATS*. Springer, 121–130.

[24] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. 2013. Discovering Denial Constraints. *PVLDB* 6, 13 (2013), 1498–1509.

[25] Diego Colombo and Marloes H. Maathuis. 2014. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.* 15, 1 (2014), 3741–3782.

[26] Rachel Cummings, Sara Krehbiel, Kevin A. Lai, and Uthaipon Tao Tantipongpipat. 2018. Differential Privacy for Growing Databases. In *NeurIPS*. 8878–8887.

[27] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[28] Cynthia Dwork. 2006. Differential Privacy. In *ICALP*, Vol. 4052. Springer, 1–12.

[29] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT*, Vol. 4004. Springer, 486–503.

[30] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography (TCC '06)*. 265–284.

[31] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. 2009. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*. ACM, 381–390.

[32] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[33] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proc. VLDB Endow.* 11, 11 (2018), 1454–1467.

[34] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *CCS*. ACM, 1054–1067.

[35] Ju Fan, Tongyu Liu, Guoliang Li, Junyou Chen, Yuwei Shen, and Xiaoyong Du. 2020. Relational Data Synthesis using Generative Adversarial Networks: A Design Space Exploration. *Proc. VLDB Endow.* 13, 11 (2020), 1962–1975.

[36] Liyue Fan. 2020. A Survey of Differentially Private Generative Adversarial Networks. In *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*.

[37] Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. 2011. Discovering Conditional Functional Dependencies. *IEEE Trans. Knowl. Data Eng.* 23, 5 (2011), 683–698.

[38] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. 2019. Differentially Private Generative Adversarial Networks for Time Series, Continuous, and Discrete Open Data. In *SEC*. 151–164.

[39] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. 2014. Dual Query: Practical Private Query Release for High Dimensional Data. In *ICML*, Vol. 32. 1170–1178.

[40] Chang Ge, Xi He, Ihab F. Ilyas, and Ashwin Machanavajjhala. 2019. APEx: Accuracy-Aware Differentially Private Data Exploration. In *SIGMOD*. 177–194.

[41] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F. Ilyas. 2020. Kamino: Constraint-Aware Differentially Private Data Synthesis. arXiv:2012.15713 [cs.DB]

[42] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. *CoRR* abs/1406.2661 (2014).

[43] Andy Greenberg. 2016. Apple's 'Differential Privacy' Is About Collecting Your Data—But Not *Your* Data. *Wired* (2016).

[44] Rahul Gupta. 2019. Data Augmentation for Low Resource Sentiment Analysis Using Generative Adversarial Networks. In *ICASSP*. IEEE, 7380–7384.

[45] Michael B. Hawes. 2020. Implementing Differential Privacy: Seven Lessons From the 2020 United States Census. *Harvard Data Science Review* (30 4 2020). https://doi.org/10.1162/99608f92.353c6f99 https://hdsr.mitpress.mit.edu/pub/dgg03vo6.

[46] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *Comput. J.* 42, 2 (1999), 100–111.

[47] IBM. 2020. Cost of a Data Breach Report. (2020).

[48] Ihab F. Ilyas and Xu Chu. 2019. *Data Cleaning*. ACM.

[49] Noah M. Johnson, Joseph P. Near, and Dawn Song. 2018. Towards Practical Differential Privacy for SQL Queries. *PVLDB* 11, 5 (2018), 526–539.

[50] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. 2019. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *ICLR*.

[51] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

[52] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press.

[53] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. PrivateSQL: A Differentially Private SQL Query Engine. *PVLDB* 12, 11 (2019), 1371–1384.

[54] Warner S. L. 1965. Randomized response: a survey technique for eliminating evasive answer bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–66.

[55] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. 2015. The matrix mechanism: optimizing linear counting queries under differential privacy. *VLDB J.* 24, 6 (2015), 757–781.

[56] Haoran Li, Li Xiong, Lifan Zhang, and Xiaoqian Jiang. 2014. DPSynthesizer: Differentially Private Data Synthesizer for Privacy Preserving Data Sharing. *Proc. VLDB Endow.* 7, 13 (2014), 1677–1680.

[57] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *ICML*, Vol. 97. 4435–4444.

[58] Frank McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul (Eds.). ACM, 19–30.

[59] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.

[60] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 263–275.

[61] National Institute of Standards and Technology. 2018. Differential Privacy Synthetic Data Challenge. https://www.nist.gov/ctl/pscr/open-innovation-prize-challenges/past-prize-challenges/2018-differential-privacy-synthetic

[62] Art B. Owen. 2013. *Monte Carlo theory, methods and examples.*

[63] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. In *ICLR*.

[64] Eduardo H. M. Pena, Eduardo Cunha de Almeida, and Felix Naumann. 2019. Discovery of Approximate (and Exact) Denial Constraints. *Proc. VLDB Endow.* 13, 3 (2019), 266–278.

[65] Haoyue Ping, Julia Stoyanovich, and Bill Howe. 2017. DataSynthesizer: Privacy-Preserving Synthetic Datasets. In *SSDBM*. ACM, 42:1–42:5.

[66] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. 2014. PriView: practical differentially private release of marginal contingency tables. In *SIGMOD*. 1435–1446.

[67] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *PVLDB* 10, 11 (2017), 1190–1201.

[68] Matthew Richardson and Pedro M. Domingos. 2006. Markov logic networks. *Machine Learning* 62, 1-2 (2006), 107–136.

[69] Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. 2019. A Formal Framework for Probabilistic Unclean Databases. In *ICDT*. 6:1–6:18.

[70] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. 2017. Learning from Simulated and Unsupervised Images through Adversarial Training. In *CVPR*. IEEE Computer Society, 2242–2251.

[71] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *GlobalSIP*. 245–248.

[72] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. 2011. *Probabilistic Databases*. Morgan & Claypool Publishers.

[73] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. 2020. DP-CGAN: Differentially Private Synthetic Data and Label Generation. *CoRR* abs/2001.09700 (2020).

[74] Alexandre B. Tsybakov. 2009. *Introduction to Nonparametric Estimation*. Springer.

[75] Jonathan Ullman and Salil P. Vadhan. 2011. PCPs and the Hardness of Generating Private Synthetic Data. In *TCC*. 400–416.

[76] Christopher Waites. 2019. PyVacy: Towards Practical Differential Privacy for Deep Learning. *https://github.com/ChrisWaites/pyvacy* (2019).

[77] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Nancy Xin Ru Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. CORD-19: The COVID-19 Open Research Dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*. Association for Computational Linguistics, Online. https://www.aclweb.org/anthology/2020.nlpcovid19-acl.1

[78] Oliver Williams and Frank McSherry. 2010. Probabilistic Inference and Differential Privacy. In *NIPS*. 2451–2459.

[79] Richard Wu, Aoqian Zhang, Ihab F. Ilyas, and Theodoros Rekatsinas. 2020. Attention-based Learning for Missing Data Imputation in HoloClean. In *MLSys*.

[80] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2011. Differential Privacy via Wavelet Transforms. *IEEE Trans. Knowl. Data Eng.* 23, 8 (2011), 1200–1214.

[81] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially Private Generative Adversarial Network. *CoRR* abs/1802.06739 (2018).

[82] Sandeep Yaramakala and Dimitris Margaritis. 2005. Speculative Markov Blanket Discovery for Optimal Feature Selection. In *ICDM*. 809–812.

[83] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2014. PrivBayes: private data release via bayesian networks. In *SIGMOD*. 1423–1434.

[84] Tianqing Zhu, Gang Li, Wanlei Zhou, and Philip S. Yu. 2017. Differentially Private Data Publishing and Analysis: A Survey. *IEEE Trans. Knowl. Data Eng.* 29, 8 (2017), 1619–1638.