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 personal 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.

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 [78]. 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 [6, 45] and penalties [1]. Differential privacy (DP) [24] has emerged as a standard data privacy guarantee by government agencies [5, 42] and companies [31, 40, 47]. 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”.

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 [5, 15], building prediction models [4, 64], answering SQL queries [38, 47, 53, 58]). For example, APEx [38] 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 [15]. 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 [29].

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 [13, 28, 36, 76] 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., [8, 17, 18, 48, 84]) 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 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 [23] consisting of 15 attributes with denial constraints [46], 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 [84], PATE-GAN [48], and DP-VAE [17] have up to 32% of the tuple pairs failing these constraints (Table 2).

However, naively repairing the incorrect structure constraints in the synthetic data can compromise the usefulness. We applied state-of-the-art data cleaning method [68] 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

Therefore, 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) [46], and we present KAMINO, a system for constraint-aware differentially private data synthesis.

Our solution is built on top of the probabilistic database framework [70, 73], which models a probability distribution over ordinary databases and incorporates the denial constraints as parametric factors. Database instances that share similar structural and statistical correlations with the true data are modeled to have similar probability (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.

We consider a relational database schema of a single relation \( R = \{A_1, \ldots, A_k\} \) with \( k \) attributes. Let \( D \) be a database instance of this schema \( R \) and consist of \( n \) tuples \( \{t_1, \ldots, t_n\} \). 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 \( D(A_j) \). Index 1 refers to the first element in a list/array.

2 PRELIMINARIES

We consider a relational database schema of a single relation \( R = \{A_1, \ldots, A_k\} \) with \( k \) attributes. Let \( D \) be a database instance of this schema \( R \) and consist of \( n \) tuples \( \{t_1, \ldots, t_n\} \). 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 \( D(A_j) \). Index 1 refers to the first element in a list/array.

2.1 Denial Constraints

Denial constraints (DCs) [46] are used in practice by domain experts to specify the structure of the data, such as functional dependency (FD) [44] and conditional FD [34]. In case of missing DCs, recent work has designed algorithms to automatically discover DCs from the database instance [12, 21].

We express a DC as a first-order formula in the form of \( \phi : \forall t_1, t_2, \ldots \in D, \neg(P_1 \land \cdots \land P_m) \). Each predict \( P_i \) is of the form \( (v_1 \circ v_2) \) or \((v_1 \circ c)\), where \( v_1, v_2 \in t_i[A], x \in \{i, j, \cdots\}, A \in R, o \in \{=, \neq, \geq, \leq\} \), and \( c \) is a constant. We will omit universal quantifiers \( \forall t_1, t_2, \ldots \) hereafter for simplicity.

Example 2: Consider a database instance \( D \) with schema \( R = \{age, edu_num, edu, cap_gain, cap_loss\} \), and three DCs:

\[
\phi_1: \neg(t_i[age] = t_j[age] \land t_i[edu_num] = t_j[edu_num])
\]
\[
\phi_2: \neg(t_i[cap_gain] > t_j[cap_gain] \land t_i[cap_loss] < t_j[cap_loss])
\]
\[
\phi_3: \neg(t_i[age] < 10 \land t_i[cap_gain] > 10)
\]

The first DC \( \phi_1 \) expresses an FD \( \text{edu} \rightarrow \text{edu_num} \). It states that for any two tuples with same \( \text{edu} \), their \( \text{edu_num} \) must be the same too. The second DC \( \phi_2 \) states that for any two tuples, if one’s \( \text{cap_gain} \) is greater than the other’s, its \( \text{cap_loss} \) cannot be smaller.

Kamino was the planet in Star Wars, renowned for the technology of clone armies.
The third DC $\phi_3$ 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 $\phi$ in a database instance $D$. We refer to DC $\phi$ 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 $D \subset D$, $V(\phi, D) \subset V(\phi, D')$. We also use $A_\phi$ to represent the set of attributes that participate in the DC $\phi$. For example, $A_\phi = \{edu, edu_num\}$.

### 2.2 Probabilistic Database

To model databases that do not fully comply with a given set of DCs, we use a probabilistic database [70], a probability distribution over ordinary databases. Intuitively, a database instance with few violations is more likely: Given a set of DCs $\Phi$ and their weights $\{w_\phi \mid \phi \in \Phi\}$, the probability of an instance $D$ is defined as follows:

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

where $\prod_{t \in D} Pr(t)$ models a tuple-independent probabilistic database [70, 73], wherein each tuple independently comes from a probability distribution over tuples, and $|V(\phi, D)|$ is the size of violations of DC $\phi$ on $D$. Each DC $\phi$ is associated with a weight $w_\phi$, and each violation of $\phi$ 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 $w_\phi$. 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 [70].

### 2.3 Tuple Embedding

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

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

Each conditional probability is learned as a discriminative model based on *tuple embedding* [80] and *attention mechanism* [7]. 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 [30, 80].

Consider the discriminative model used in AimNet [80] that predicts the value of target attribute $A_j$ based on the values of a set of context attributes $\{A_1, \ldots, A_{j-1}\}$. AimNet transforms each attribute value into a vector embedding with fixed dimension $d$. For an attribute with continuous values $z \in \mathbb{R}^d$, where $d' < d$, AimNet first standardizes each dimension to zero mean and unit variance, and then apply a linear layer followed by a non-linear ReLU layer to obtain a non-linear transformation of the input $z = B_0 + (A_j + c) \odot \hat{d}$, where $A$, $B$, $c$, $\hat{d}$ are learned parameters and $\odot$ is a ReLU. For each attribute with discrete values, AimNet associates it with a learnable lookup table mapping embeddings to domain values.

AimNet relies on the attention mechanism [7] to learn structural dependencies between different attributes of the input data and uses the attention weights to combine the representations of inputs into an vector representation (the context vector) for the target attribute. To predict a target attribute value, it learns the transformation from context vector back to a value in the domain of the target attribute. The output of a discriminative model is the learned representation of all the attributes, and a list of prediction probabilities for all values of a target attribute with the discrete domain, or the regression parameters (mean and std) of a Gaussian distribution for a target attribute with a continuous domain.

### 2.4 Differential Privacy

Differential privacy (DP) [26, 29] is used as our measure of privacy.

**Definition 1 (Differential Privacy (DP) [29]).** A randomized algorithm $M$ achieves $(\epsilon, \delta)$-DP if for all $S \subseteq \text{Range}(M)$ and for any two database instances $D, D' \in D$ that differ only in one tuple:

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

The privacy cost is measured by the parameters $(\epsilon, \delta)$. 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 [25] states that for any function $g$ defined over the output of the mechanism $M$, if $M$ satisfies $(\epsilon, \delta)$-DP, so does $g(M)$; 2) Composability [24] states that if $M_1, M_2, \ldots, M_k$ satisfy $(\epsilon_1, \delta_1), (\epsilon_2, \delta_2), \ldots, (\epsilon_k, \delta_k)$-DP, then a mechanism sequentially applying $M_1, M_2$ to $M_k$ satisfies $(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i)$-DP.

Gaussian mechanism [29] is a widely used DP algorithm. Given a function $f : D \rightarrow \mathbb{R}^d$, the Gaussian mechanism adds noise sampled from a Gaussian distribution $N(0, S_f^2 \sigma^2)$ to each component of the query output, where $\sigma$ 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 $\sigma \geq \sqrt{2\ln(1/\delta)/\epsilon}$, then the Gaussian mechanism satisfies $(\epsilon, \delta)$-DP.

Gaussian mechanism has been applied to answer counting queries [55]. It also been used in differentially private stochastic gradient descent (DP-SGD) [4, 10, 72, 79]. 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 $C$.

**Semantics of DP.** Prior work [9, 27, 37, 43, 49, 50, 67] offer semantic interpretations of DP with respect to adversarial knowledge: informally, regardless of external knowledge, an adversary with access to the sanitized database draws the same conclusions whether or not one’s data is included in the original database. Ganta et al. [37] formalize the notion of “external knowledge”, and of “drawing conclusions” respectively via (i) a prior probability distribution $b[\cdot]$ on the database domain $D$ of size $n$; and (ii) the corresponding posterior probability distribution: given a transcript $t$ outputted by mechanism $M$, the adversary updates his belief about the database $D$ using Baye’s
rule to obtain a posterior $\hat{b}[D|t] = \frac{\Pr[M(D|t) | b[D]]}{\sum_{b'} \Pr[M(D|t) | b'[D]]}$. Consider the hypothetical scenario where person $i$’s data is not used, denoted by $M(D_{-i})$, given a transcript $t$, the updated posterior belief about the database $D$ is defined as $\hat{b}_i[D|t] = \frac{\Pr[M(D_{-i}) | b[D]]}{\sum_{b'} \Pr[M(D_{-i}) | b'[D]]}$. A DP mechanism $M$ with proper privacy parameters (sufficiently small $\delta$) can prevent the adversary from drawing different conclusions about whether or not person $i$’s data was used, i.e., the statistical difference between $\hat{b}_i[|t]$ and $\hat{b}_i[|t]$ is small for $D$ drawn from $b[|t]$ and $t$ drawn from $M(D)$ with a high probability [37, 49]. This posterior-to-posterior comparison applies to arbitrary prior of adversaries, unlike prior-to-posterior approaches [27, 50].

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 $\Phi$ with information about their hardness, and a differential privacy budget ($\epsilon, \delta$), 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 $\Phi$ 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 $(\epsilon, \delta)$-differential privacy: for any set of output instances $\mathcal{D}$ outputted by $P$, $\Pr(P(D_1) \in \mathcal{D}) \leq e^\epsilon \Pr(P(D_2) \in \mathcal{D}) + \delta$, for any two neighboring $D_1$ and $D_2$ differing in one record.

DC constraints $\Phi$ are public in our problem and can be modeled as part of the adversary’s prior. This subsumes the special case when $\Phi$ are not public to the adversary. The semantic privacy results by Ganta et al. [37, 49] (§ 2.4) are applicable to our problem and prior work on DP data synthesis, and hence this work will focus on the design of a DP mechanism. We will leave the mechanism design for stronger semantic privacy guarantees to future work.

Naive solutions fail either R1 or R2, or both. If the privacy guarantee (R2) is not mandatory, then releasing the private database instance $D^*$ would suffice for data consistency (R1). On the contrary, if privacy is the only goal, then outputting an empty dataset will suffice. As shown in Example 1, existing practical work on private data synthesis [17, 48, 84] do not preserve DCs. A post-cleaning step can improve the data consistency, but it is at the cost of the usefulness of the synthetic for other applications (Figure 1). To the best of our knowledge, there is no existing work to address both the data consistency and privacy guarantee in the synthesis process.

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 [69, 73], and approximate methods such as gradient descent may not converge to a global optimum [70], 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 [13, 28, 36, 76] 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. Then we apply Markov Chain Monte Carlo (MCMC) sampling technique [63] to obtain an instance based on the learned conditional probabilities. This sampling process approximates the true database instance probability in Eqn. (1), and hence the synthesized instance has similar properties as $D^*$.  

**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 $\phi$, 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 $D_j = \{t_1, t_2, \ldots, t_{i-1}\}$ (with $D_1 = \emptyset$) from $D$, for $i = 1, \ldots, n$. Let $V(\phi, t_i | D_i)$ denote the set of new violations caused by tuple $t_i$ with respect to $D_i$. Then we have

$$V(\phi, D) = V(\phi, t_1) + V(\phi, t_2 | D_2) + \cdots + V(\phi, t_n | D_n)$$

(3)

This allows us to decompose Eqn. (1) as

$$\Pr(D) \propto \prod_{i=1}^{n} \Pr(t_i) \times \exp \left( - \sum_{\phi \in \Phi} w_{\phi} \sum_{i=1}^{n} |V(\phi, t_i | D_i)| \right)$$

$$= \prod_{i=1}^{n} \Pr(t_i) \times \exp \left( - \sum_{\phi \in \Phi} w_{\phi} \times |V(\phi, t_i | D_i)| \right)$$

(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 $j$th attribute in $S$ and $S_1 = \emptyset$ for the purpose of uniform representation. This schema sequence allows us to further decompose the set of violations. Let $\Phi_{S_j}$ represent the set of DCs in $\Phi$ 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 $S = [\text{age, edu, num, edu, cap_gain, cap_loss}]$, we can verify that $\Phi_{S_1} = \{\phi_1\}$, as the attributes $\{\text{edu, num, edu}\}$ for $\phi_1$ are covered by the first 3 attributes in $S$, but not the first 2 attributes.

Notice that for a DC $\phi \in \Phi_{S_j}$, given a tuple $t_i$, its number of violations $|V(\phi, t_i | D_i)|$ only depends on the values of the first $j$
Figure 2: Sampling values in an instance (Example 4).

Algorithm 1 Constraint-aware differentially private data synthesis

Require: Private instance $D'$, domain $D$
Require: DCs $\Phi$, privacy budget $(\epsilon, \delta)$

1. $S \leftarrow$ SEQUENCING$(R, D, \Phi)$ \rightarrow Algorithm 4
2. $\Psi \leftarrow$ SEARCHDParas$(\epsilon, \delta, D, S)$ \rightarrow Algorithm 6
3. $M \leftarrow$ TRAINMODEL$(D^*, S, D, \Psi)$ \rightarrow Algorithm 2
4. $W \leftarrow$ LEARNWEIGHT$(D', \Psi, S, M, \Psi)$ \rightarrow Algorithm 5
5. $D' \leftarrow$ SYNTHESIZE$(S, M, \Psi, D, W)$ \rightarrow Algorithm 3
6. return $D'$
7. end procedure

The table contains data on attributes such as age, education level, and cap_gain. It appears to be a sample dataset for a study on educational outcomes.

Algorithmic description (continued):

Algorithm 1 Constraint-aware differentially private data synthesis

Require: Private instance $D'$, domain $D$
Require: DCs $\Phi$, privacy budget $(\epsilon, \delta)$

1. $S \leftarrow$ SEQUENCING$(R, D, \Phi)$ \rightarrow Algorithm 4
2. $\Psi \leftarrow$ SEARCHDParas$(\epsilon, \delta, D, S)$ \rightarrow Algorithm 6
3. $M \leftarrow$ TRAINMODEL$(D^*, S, D, \Psi)$ \rightarrow Algorithm 2
4. $W \leftarrow$ LEARNWEIGHT$(D', \Psi, S, M, \Psi)$ \rightarrow Algorithm 5
5. $D' \leftarrow$ SYNTHESIZE$(S, M, \Psi, D, W)$ \rightarrow Algorithm 3
6. return $D'$
7. end procedure

The system overview describes the process of generating a synthetic database instance. Kamino first chooses a schema sequence $S$ based on the schema $R$, domain $D$, and DCs $\Phi$ (Line 2). Then it finds a suitable parameter set $\Psi$ for the subsequent algorithms to ensure the overall privacy loss is bounded by $(\epsilon, \delta)$-DP (Line 3). The algorithms TRAINMODEL(-) and LEARNWEIGHT(-) 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.

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 Eqn. (2) that, given a schema sequence $S = [A_1, A_2, \ldots, A_k]$, the tuple probability becomes $Pr[t] = Pr[(t[A_1]) \cdot \prod_{i=2}^k Pr[t[A_i] \mid t[A_1, \ldots, A_{i-1}]]]$. 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 $Pr[t[A_i] \mid t[A_1, \ldots, A_{i-1}]]$, we train a discriminative sub-model that uses context attributes $(A_1, \ldots, A_{i-1})$ to predict the target attribute $A_i$. We denote this sub-model by $M_X$, 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
Algorithm 2 Probabilistic data model training

Require: \( D^*, D, S \) \rightarrow True instance, domain, schema sequence
Require: \( n, k, \eta, q \) \rightarrow cardinality, dimensions, lr, quantization
Require: \( \sigma_p, \sigma_d \) \rightarrow Noise scales in \( \Psi \)
Require: \( C, T, b \) \rightarrow L_2 norm clip/#iterations/batch size in \( \Psi \)

1: procedure TrainModel\((D^*, D, S)\) \( \Psi \)
  2: \( H \leftarrow \) counts of (quantized) values in \( D^* \) for 1st attr. \( S[1] \)
  3: Add noise drawn from \( N(0, 2\sigma_d^2) \) to each count in \( H \)
  4: \( M_{0,S[1]} \leftarrow \) 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 \( \rightarrow \) Context attributes
    8: \( y = S[j] \), initialize embedding \( \rightarrow \) Target attribute
    9: Initialize discriminative model \( M_{X,y} \) \( \rightarrow \) [80]
    10: \( L(\theta_y, t) \leftarrow \) loss function on imputing target \( y \)
    11: for \( e \in [T] \) do
        12: \( D_e \leftarrow \) random sample on \( D^*[X, y] \) with prob \( b/n \)
        13: For each \( t \in D_e \), compute \( g_e(t) \leftarrow \nabla_{\theta_y} L(\theta_y, t) \)
        14: \( g_e(t) \leftarrow \max(1, \frac{\|g_e(t)\|}{\epsilon}) \) \( \rightarrow \) Clip gradient
        15: \( g_e \leftarrow (\sum_{t \in D_e} g_e(t) + N(0, \sigma^2 C^2_\epsilon))/b \) \( \rightarrow \) Add noise
        16: \( \theta_y \leftarrow \theta_y - \eta \times g_e \) \( \rightarrow \) 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 \( M \)
end procedure

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 \( M_{0,S[1]} \), and the parameters in the \((k-1)\) discriminative sub-models \( M_{S[j],S[1]} \) for \( j \in [2, k] \). It takes as input of the true database instance \( D^* \) with domain \( D \), the schema sequence \( S \) (to be discussed in § 4.3), as well as learning parameters (number of iterations \( T \), batch size \( b \), learning rate \( \eta \), and quantizing \( q \) bins for numerical attributes) and noise parameters (\( \sigma_p \) and \( \sigma_d \) for Gaussian noise, \( L_2 \) norm clip threshold for gradients \( C \)).

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 target attribute \( A_j \) is a continuous domain, the conditional probability \( p_{\phi}(t_j | S[j]) \) predicts the target attribute \( A_j = \phi \) given the context attributes \( S[j] = c \). If the target attribute \( A_j \) is a continuous domain, the discriminative model is based on regression model and outputs a Gaussian distribution mean \( m \) and std \( \sigma \), given the context attributes \( S[j] = c \).

We sample \( t_j \) from the distribution and assign each candidate \( \phi \) with a probability \( p_{\phi} \propto \exp(-\frac{1}{\sigma^2} \exp(-\frac{1}{\epsilon^2}(\frac{m-c}{\sigma})^2)). \) The other values in the domain are assigned with probability 0. We denote the candidate set by \( D(S[j]) \).

Next, we compute the number of DC violations \( \text{vio}_{\phi, i | D^*} \) if we assign \( t_i | A_j = \phi \):

\[ |\{V(\phi, t_i | S[j]) = c \cap t_i | A_j = \phi \ | D^*[S[j+1]]\}| \]

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

\[ P(\phi) \propto p_{\phi} \cdot \exp(-\sum_{\phi \in \Phi_{A_j}} \text{vio}_{\phi, i | D^*}) \]

and update the \( j \)th attribute of \( t_i \) (Line 10). Once we fill up all the values for attribute \( A_j \), we re-sample values of the same attribute 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).

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.
**Algorithm 3 Constraint-aware database instance sampling**

**Require:** \(S, M, \Phi, D\) Schema sequence, data model, DCs, domain

**Require:** \(W, L, N\) → Weight vector (Alg. 5), sample size, #round

1: **procedure** SYNTHESIZE\((S, M, \Phi, D)\)
2: \[D'[S[1]] \leftarrow \text{sample from distribution } M_{0,S[1]}\]
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: \[p_{\phi}[c] \mid v \in D(S[j]) \leftarrow M_{S,i,c,S[j]}\]
7: for \(v \in D(S[j])\) and \(\phi \in \Phi_{S[j]}\) do
8: \(\phi|_{\phi,v,D'} \leftarrow \text{num. of vio. of } \phi \text{ if } t_i[S[j]] = v\)
9: end for
10: Update \(t_i[S[j]] = v\) where \(v\) is sampled with \(P[v] \propto e^{-\sum_{\phi\in\Phi_{S[j]}} w_\phi \times \phi|_{\phi,v,D'}}\)
11: end for
12: Re-sample till convergence or a fixed number of rounds
13: end for
14: return \(D'\)
15: **end procedure**

\(S[j]\) (Line 12), by randomly choosing a cell \(t_i[A_j]\) and re-sampling its values conditioned on all other values that have been generated in \(D'\) so far. The final output is a synthetic database instance \(D'\) of size \(n\) with the same schema as the true database instance \(D^n\).

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., \(\phi_1\)) or single-tuple DCs (e.g., \(\phi_2\)), because not all the domain values appear in the true data \(D^n\). 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. General MCMC sampling (see Example 4) would require re-sampling on the entire full \(D^n\) with all attributes, and hence \(k-1\) more conditional distributions need to be learned. However, in the privacy setting with a fixed privacy budget, learning more distributions will compromise the accuracy of each learned distribution, and hence MCMC sampling may converge slow. Therefore, Kamino applies a constrained MCMC sampling for each attribute, following the given schema sequence (§ 4.3).

### 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 [22, 83] that spend part of the privacy budget in learning a good sequence, we make use of the input DCs \(\Phi\) and the domain \(D\). This heuristic approach incurs no privacy cost since the true database instance \(D^n\) is not queried.

Specifically, we propose a rule-based, instance-independent method to ensure that for an FD \(X \rightarrow Y\) in \(\Phi\), 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 \(\Sigma\) by the minimal domain size of an attribute from \(X\) (i.e., \(\exists A^1 \in X_1, \forall A^2 \in X_2, |D(A^1)| \leq |D(A^2)|\)) (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 overall complexity is linear with respect to the number of attributes.

Our sequencing algorithm relies on the given FDs as a subset of DCs. In cases that \(\Phi\) 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 (\(2\times3\times5\)), instead of 20 on the reversed sequence.

#### 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 [84] 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 [21] 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 Learning DC weights

Require: \( D^*, \Phi, S \) → True instance, DCs, schema sequence
Require: \( \sigma_w, T_w, L_w \) → Noise scale/#iteration/sample size in \( \Psi \)
Require: \( b_w, S_w \) → Batch size in \( \Psi \), sensitivity (Lemma 1)

```
procedure LearnWeight(D^*, \Phi, S, M, \Psi)
1: Initialize weight vector \( W \) of length \(|\Phi|\) if unknown
2: Take a random sample \( \tilde{D} \) from \( D^* \) with a probability \( L_w/n \)
3: Drop tuples from the sample if \(|\tilde{D}| > L_w \)
4: Compute violation matrix \( V \) of size \((|\tilde{D}| \times |\Phi|)\) from \( \tilde{D} \)
5: Add noise drawn from \( \mathcal{N}(0, S_w^2 \sigma_w) \) to each value in \( V \)
6: Set negative values in \( V \) to zero
7: for \( A_j \in S \) and \( \epsilon \in [T_w] \) do
8: \hspace{1em} ids ← sample \( b \) ids from \([1, L_w]\) with prob \( b_w/L_w \)
9: \hspace{2em} for each \( i \) in ids do
10: \hspace{3em} \( O ← \exp(-\sum_{\phi_j \in \Phi_{A_j}} W[i] \cdot V[i][i]) \)
11: \hspace{2em} Update \( W \) via back propagation by max \( O \)
12: end for
13: end for
14: return \( W \)
15: end procedure
```

Algorithm 6 Searching DP parameters

Require: \( \epsilon, \delta, \mathcal{D}, S \) → Privacy budget, domain, schema sequence

```
procedure SearchDPPara(\epsilon, \delta, \mathcal{D}, S)
1: \( C ← 1, \sigma_d ← 1.1, \eta ← 10^{-4} \) → norm clip, noise scale, lr
2: \( \sigma_e \in [0.1/|\mathcal{D}(\mathcal{S}[1])|, 4\sqrt{\log(1.25/\delta)}/\epsilon, \sigma_d \in [1, 1.5] \)
3: \( b \in [16, 32], T \in [n/min(b), 5n/min(b)] \)
4: Initialize \( \sigma_p, \sigma_d \) to the minimal, and \( T, b \) to the maximal
5: if DC weights unknown then
6: \hspace{1em} \( \epsilon_w, L_w ← 100, \sigma_w ← \sqrt{2\log(1.25/\delta_w)/\epsilon_w} \)
7: \hspace{1em} \( b_w ← 1, T_w ← L_w/b_w \)
8: end if
9: while \( \epsilon_p(\delta) > \epsilon \) do
10: \hspace{1em} If \( T > T_{\text{min}} \), then decrease \( T \)
11: \hspace{1em} If \( \sigma_d < \sigma_{d_{\text{max}}} \), then increase \( \sigma_d \)
12: \hspace{1em} If \( \sigma_e < \sigma_{e_{\text{max}}} \), then increase \( \sigma_e \)
13: \hspace{1em} If \( b > b_{\text{min}} \), then decrease \( b \)
14: end while
15: return \( \Psi \), a set consisting of all above parameters
16: end procedure
```

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.

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) do not access the true database instance. Hence, we can show KAMINO achieves DP by simple sequential composition [24] and post-processing property [25] of DP. However, this does not give us the tightest privacy bound. Instead, we apply R\"{e}nyi DP (RDP) [60], a generalized privacy notion and its advanced composition techniques for the privacy analysis of KAMINO.

Definition 2 (R\"{e}nyi-DP [60]). A randomized algorithm \( M \) with domain \( \mathcal{D} \) is \((a, \epsilon)\)-RDP at order \( \alpha > 1 \), for any pair of neighboring database instances \( D, D' \in \mathcal{D} \) that differ in one tuple. Let \( P_D \) and \( P_{D'} \) be the output probability density of \( M(D) \) and \( M(D') \), respectively. It holds that:

\[
\frac{1}{\alpha-1} \log \mathbb{E}_{x \sim \mathcal{M}(D')} \left( \frac{P_x(x)}{P_{D'}(x)} \right)^\alpha \leq \epsilon.
\]

Both the post-processing and composability properties apply to RDP. Specifically, if a sequence of adaptive mechanisms \( M_1, M_2, \ldots, M_k \) satisfy \((a, \epsilon_1)\)-, \((a, \epsilon_2)\)-, \ldots, \((a, \epsilon_k)\)-RDP, then the composite privacy loss is \((a, \sum_{i=1}^k \epsilon_i)\)-RDP.
As we applied the Gaussian mechanism on sampled data, we summarize the RDP privacy loss of a generalized mechanism, the sampled Gaussian mechanism (SGM) [61].

**Lemma 2.** Given a database $D$ and query $f: D \rightarrow \mathbb{R}^d$, returning
$f(\{x \in D \mid x \text{ is sampled with probability } r\}) + \mathcal{N}(0, S^2 \sigma^2)$ results in the following RDP cost for an integer moment $\alpha$

$$R_{\alpha, r}(\alpha) = \left\{ \begin{array}{ll}
\frac{\alpha}{2 \sigma^2} + \sum_{k=0}^{\alpha} \left(1 - \frac{b}{n} \right)^{\alpha-k} \left(1 - \frac{b}{n} \right)^k \exp\left(\frac{\alpha^2 - \alpha}{2 \sigma^2}\right) & r = 1 \\
\sum_{k=0}^{\alpha} \left(1 - r\right)^{\alpha-k} \exp\left(\frac{\alpha^2 - \alpha}{2 \sigma^2}\right) & 0 < r < 1
\end{array} \right.$$

We analyze the RDP cost of each step in KAMINO and result in the following total cost.

**Theorem 1.** The total RDP cost of KAMINO with parameter configuration set $\Psi = \{\sigma, \sigma_b, \sigma_w, b, T, k, L_w, i_w\}$ (Algorithm 1) is

$$R_{\Psi}(\alpha) = \frac{\alpha}{2 \sigma^2} + T(k - 1) \times \sum_{k=0}^{\alpha} \left(1 - \frac{b}{n} \right)^{\alpha-k} \left(1 - \frac{b}{n} \right)^k \exp\left(\frac{\alpha^2 - \alpha}{2 \sigma^2}\right).$$

where $i_w$ is a binary indicator for $M_3$ (DC weight learning).

**Proof.** KAMINO has the following adaptive SGMs:

- For $M_1$, the sampling rate is 1, $R_{M_1}(\alpha) = \alpha/2 \sigma^2$.
- For $M_2$, the sampling rate is set to $b/n$, and SGM is applied $T \times (k - 1)$ times. Thus, $R_{M_2}(\alpha) = T(k - 1) \times \sum_{k=0}^{\alpha} \left(1 - \frac{b}{n} \right)^{\alpha-k} \left(1 - \frac{b}{n} \right)^k \exp\left(\frac{\alpha^2 - \alpha}{2 \sigma^2}\right)$.
- For $M_3$, the sampling rate is $L_w/n$. Thus, $R_{M_3}(\alpha) = \sum_{k=0}^{\alpha} \left(1 - \frac{L_w}{n} \right)^{\alpha-k} \exp\left(\frac{\alpha^2 - \alpha}{2 \sigma^2}\right)$.

By the composition property [60] of RDP, the total RDP cost is:

$$R_{KAMINO}(\alpha) = \frac{\alpha}{2 \sigma^2} + \sum_{k=0}^{\alpha} \left(1 - \frac{L_w}{n} \right)^{\alpha-k} \left(1 - \frac{L_w}{n} \right)^k \exp\left(\frac{\alpha^2 - \alpha}{2 \sigma^2}\right) + T(k - 1) \times \sum_{k=0}^{\alpha} \left(1 - \frac{b}{n} \right)^{\alpha-k} \left(1 - \frac{b}{n} \right)^k \exp\left(\frac{\alpha^2 - \alpha}{2 \sigma^2}\right).$$

By the tail bound property of RDP [60], we can convert the RDP cost of KAMINO to $(\epsilon, \delta)$-DP, where $\epsilon$ is computed by

$$\epsilon_{\Psi}(\delta) = \min_k R_{\Psi}(\alpha) + \frac{\log(1/\delta)}{\alpha - 1}.$$  

(7)

for a given $\delta$. The order $\alpha$ is usually searched within a range [77].

In practice, the overall privacy budget $(\epsilon, \delta)$ is specified as an input to KAMINO, and one needs to judiciously set the privacy parameters in $\Psi$. 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 $(\epsilon, \delta)$ and outputs a set of parameters $\Psi$ that ensures that the overall privacy cost does not exceed $(\epsilon, \delta)$. It starts with a default setting based on prior experimental heuristics [11, 77] and the domain information $\mathcal{D}$. The noise parameters including $(\sigma, \sigma_b, \sigma_w, b, T, L_w)$ are boldly set to give the best possible accuracy (Line 5). If this privacy cost of this configuration is higher than $\epsilon$ (Line 10), then we use a priority order to decide which parameter to tune (Lines 11-14). This process is repeated till the privacy cost is capped at our total budget.

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 $\omega$-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.

7.1 Evaluation Setup

Datasets. We choose 4 different datasets with mixed data types and DCs, listed in Table 1. First, the Adult dataset [23] consists of 15 census attributes and 2 hard DCs. Second, the BR2000 dataset [84] has a smaller domain size than the Adult dataset, but it has 3 soft DCs with unknown weights. The third dataset, Tax [21], has a very large domain size, e.g., zip ($= 215$) and city ($= 214$) and 6 hard DCs. Last, TPC-H [3], 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 [84], a statistical method based on Bayesian network; 2) PATE-GAN [48], a GAN-based method that trains a data generator using the PATE’s student-teacher model [64]; 3) DP-VAE [17], which samples from the latent space of a privately trained auto-encoder [51]; and 4) The winning solution of the NIST challenge [62] (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 [32]. 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 DCs in Table 1 are binary, we measure the percentage of tuple pairs that violate DCs in an instance $D$ of size $n$, i.e., $100 \cdot |V(D')| / (\binom{n}{2})$. 

---

*Analysis on general fractional moments can be found in [61]*.
Table 1: Description of the datasets that are used in the experiments.

| Dataset    | n     | k     | Domain size | Hard DCs | DCs (omitting the universal quantifier) |
|------------|-------|-------|-------------|----------|----------------------------------------|
| Adult      | 32,561| 15    | ≈ 2^{32}   | Yes      | \(\phi_1: \neg t_i[\text{edu}] = t_i[\text{edu}] \land t_i[\text{edu num}] \neq t_i[\text{edu num}]\) | \(\phi_2: \neg t_i[\text{cap gain}] > t_i[\text{cap gain}] \land t_i[\text{cap loss}] < t_i[\text{cap loss}]\) |
| BR2000     | 38,000| 14    | ≈ 2^{16}   | No       | \(\phi_1: \neg t_i[a13] = t_i[a13] \land t_i[a11] < t_i[a11] \land t_i[a3] > t_i[a3]\) | \(\phi_2: \neg t_i[a12] \not\equiv t_i[a12] \land t_i[a13] \equiv t_i[a13] \geq t_i[a3]\) |
| Tax        | 30,000| 12    | ≈ 2^{21}   | Yes      | \(\phi_1: \neg t_i[\text{zip}] = t_i[\text{zip}] \land t_i[\text{city}] \neq t_i[\text{city}]\) | \(\phi_2: \neg t_i[\text{areacode}] = t_i[\text{areacode}] \land t_i[\text{state}] \neq t_i[\text{state}]\) |
| TPC-H      | 20,000| 9     | ≈ 2^{42}   | Yes      | \(\phi_1: \neg t_i[\text{castkey}] = t_i[\text{castkey}] \land t_i[\text{nationkey}] \not\equiv t_i[\text{nationkey}]\) | \(\phi_2: \neg t_i[\text{castkey}] = t_i[\text{castkey}] \land t_i[\text{mktsegment}] \not\equiv t_i[\text{mktsegment}]\) |

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 \(A\), we compute the α-way marginal, \(h : D \rightarrow \mathbb{R}^{(|D(A)|)}\) on the synthetic data \(D'\) and true data \(D\), respectively, and then report the total variation distance [75] as \(\max_{a \in D(A)} |h(D')[a] - h(D)[a]|\).

Implementation details. KAMINO was implemented in Python 3.6. For the discriminative sub-models, we integrated the code from AimNet in the Holoclean³ system. For the baselines (PrivBayes⁴, PATE-GAN⁵, DP-VAE and NIST⁶), we reused the code from their authors with all default parameters. All the 9 models in the learning task were implemented using standard libraries [2, 19] 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.

7.2 End-to-End Evaluation

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

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 \(\phi_1\) on the Adult dataset has about 11.3%, 32%, and 20.3% violations in the synthetic data generated by PrivBayes, DP-VAE and PATE-GAN, respectively. Although NIST does not have violations like KAMINO, it because NIST filled the entire \(\text{edu num}\) 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 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 between 5 and 16 hours, which is still practically efficient.

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:

- 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 $\phi^a_i : edu \rightarrow edu\_num$ 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: 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 ($\epsilon, \delta$) at different $\epsilon = [0.1, 0.2, 0.4, 0.8, 1.6]$, while keep constant $\delta = 10^{-6}$. $\epsilon = \infty$ refers to non-private KAMINO.
First of all, increasing the privacy budget leads to overall better quality in both the learning and the query task. Second, even at small privacy budget, the quality achieved by Kamino is still reasonably good, and for some cases, even better than baseline methods with a larger privacy budget. For example, the averaged model accuracy over all attributes on Kamino is 0.8 at privacy budget (0.2, 10^{-6}), which already outperforms DP-VAE (0.54), NIST(0.66), PrivBayes (0.68) and PATE-GAN (0.77) at 5x larger $\epsilon = 1$.

### 8 RELATED WORK

There has been extensive literature on releasing differentially private synthetic data [14, 33, 62, 85]. 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 [46] enforced among tuples, rather than simply assuming tuple independence.

Statistical approaches for generating synthetic data usually estimate low-dimensional marginal distributions [66, 81], due to the hardness of privatizing high-dimensional data with differential privacy guarantees [13, 28, 36, 76]. 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, 84] or undirected graphs [18, 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 [71], videos [16] and natural languages [41]. Different from unstructured data, structured data is defined using relational schema and hence, structure correlations naturally exist. Naïvely applying deep learning models such as GAN [39] 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 [32]. Second, similar to statistical approaches, methods based on deep models (e.g. [35, 48, 74, 82]) suffer from missing structure correlations.

In general, generating differentially private synthetic data is hard, due to the tradeoff between accuracy and privacy [13, 28, 36, 76]. 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., [8, 17, 18, 48, 84]) therefore choose to privately learn only a subset of correlations to model the true distribution 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.

In general, generating differentially private synthetic data is hard, due to the tradeoff between accuracy and privacy [13, 28, 36, 76]. 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., [8, 17, 18, 48, 84]) 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] Version 0.2.13. scikit-learn. Machine Learning in Python. https://scikit-learn.org/
[3] Version 2.18.0. The TPC Benchmark H (TPC-H). http://www.tpc.org/bench/tpch/.
[4] Martin 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.
[5] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In KDD. 2667.
[6] 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).
[7] Dmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In ICLR.
[8] 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.
[9] Raef Bassily, Adam Groce, Jonathan Katz, and Adam Smith. 2013. Coupled-Worlds Privacy: Exploiting Adversarial Uncertainty in Statistical Data Privacy. In Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS ’13). IEEE Computer Society, USA, 439–448. https://doi.org/10.1109/FOCS.2013.54
[10] Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. 2014. Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. In FOCS. 464–473.
[11] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. J. Mach. Learn. Res. 13 (2012), 281–305.
[12] Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. 2017. Efficient Denial Constraint Discovery with Hydra. PVLDB 11, 3 (2017), 311–323.
[13] Nirum Blum, Katrina Ligett, and Aaron Roth. 2008. A learning theory approach to non-interactive database privacy. In STOC. ACM, 69–68.
[14] 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
[15] U.S. Census Bureau. Accessed on 2020-11-30. LEHD Origin-Destination Employment Statistics 2002-2017. https://onthehmap.ces.census.gov/
[16] R. Chawla. 2019. Deepfakes : How a pervert shook the world. International Journal for Advance Research and Development 4 (2019), 4–8.
[17] Qinqing Chen, Chong Xiang, Minhui Xue, Bo Li, Nikita Borsiov, Dali Kafar, and Haojin Zhu. 2018. Differentially Private Data Generative Models. CoRR abs/1812.02274 (2018).
[18] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. 2015. Differentially Private High-Dimensional Data Publication via Sampling-Based Inference. In SIGKDD. 129–138.
[19] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In SIGKDD. ACM, 785–794.
[20] David Maxwell Chickering. 1995. Learning Bayesian Networks is NP-Complete. In AISTATS. Springer, 121–130.
[21] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. 2013. Discovering Denial Constraints. PVLDB 6, 13 (2013), 1498–1509.
[22] Diego Colombo and Marloes H. Maathuis. 2014. Order-independent constraint-based causal structure learning. J. Mach. Learn. Res. 15, 1 (2014), 3741–3782.
[23] Dhruve Dias and Casey Graf. 2017. UCI Machine Learning Repository: http://archive.ics.uci.edu/ml
[24] Cynthia Dwork. 2006. Differential Privacy. In ICALP. Vol. 4052. Springer, 1–12.
[25] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In EUROCRYPT. Vol. 4041. Springer, 486–503.
[26] 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.
[27] Cynthia Dwork and Moni Naor. 2006. On the Difficulties of Disclosure Prevention in Statistical Databases or The Case for Differential Privacy. J. Priv. Conf. Security (FOCS ’06). 1. https://doi.org/10.29012/jpc.v2i1.585
[28] 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.
[29] 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.
[30] Muhammad Ebrahimm, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. Proc. VLDB Endowe. 11, 11 (2018), 1454–1467.
[31] Ulfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In CCS. ACM, 1054–1067.
[32] Xu Fu, 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 Endowe. 13, 11 (2020), 1962–1975.
[33] Luyue Fan. 2020. A Survey of Differentially Private Generative Adversarial Networks. In The AAAI Workshop on Privacy-Preserving Artificial Intelligence.
[34] Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. 2011. Discovering Conditional Functional Dependencies. IEEE Trans. Knowl. Data Eng. 23, 5 (2011), 683–698.
[35] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Daverger. 2019. Differentially Private Generative Adversarial Networks for Time Series, Continuous, and Discrete Open Data. In SEC. 151–164.
[36] Marco Gaboardi, Emilio Jesus Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. 2014. Dual Query: Practical Private Query Release for High-
