Private Data Exploring, Sampling, and Profiling

by

Chang Ge

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Amol Deshpande Professor, Department of Computer Science University of Maryland
Supervisor:	Ihab F. Ilyas Professor, Cheriton School of Computer Science University of Waterloo
Internal Members:	Xi He Assistant Professor, Cheriton School of Computer Science University of Waterloo
	Florian Kerschbaum Associate Professor, Cheriton School of Computer Science University of Waterloo
Internal-External Member:	Helen Chen Professor of Practice, School of Public Health Sciences University of Waterloo

Author's Declaration

Abstract

Data analytics is being widely used not only as a business tool, which empowers organizations to drive efficiencies, glean deeper operational insights and identify new opportunities, but also for the greater good of society, as it is helping solve some of world's most pressing issues, such as developing COVID-19 vaccines, fighting poverty and climate change. Data analytics is a process involving a pipeline of tasks over the underlying datasets, such as data acquisition and cleaning, data exploration and profiling, building statistics and training machine learning models. In many cases, conducting data analytics faces two practical challenges. First, many sensitive datasets have restricted access and do not allow unfettered access; Second, data assets are often owned and stored in silos by multiple business units within an organization with different access control. Therefore, data scientists have to do analytics on private and siloed data.

There is a fundamental trade-off between data privacy and the data analytics tasks. On the one hand, achieving good quality data analytics requires understanding the whole picture of the data; on the other hand, despite recent advances in designing privacy and security primitives such as differential privacy and secure computation, when naïvly applied, they often significantly downgrade tasks' efficiency and accuracy, due to the expensive computations and injected noise, respectively. Moreover, those techniques are often piecemeal and they fall short in holistically integrating into end-to-end data analytics tasks.

In this thesis, we approach this problem by treating privacy and utility as constraints on data analytics. First, we study each task and express its utility as data constraints; then, we select a principled data privacy and security model for each task; and finally, we develop mechanisms to combine them into end to end analytics tasks. This dissertation addresses the specific technical challenges of trading off privacy and utility in three popular analytics tasks.

The first challenge is to ensure query accuracy in private data exploration. Current systems for answering queries with differential privacy place an inordinate burden on the data scientist to understand differential privacy, manage their privacy budget, and even implement new algorithms for noisy query answering. Moreover, current systems do not provide any guarantees to the data analyst on the quality they care about, namely accuracy of query answers. We propose APEx, a generic accuracy-aware privacy query engine for private data exploration. The key distinction of APEx is to allow the data scientist to explicitly specify the desired accuracy bounds to a SQL query. Using experiments with query benchmarks and a case study, we show that APEx allows high exploration quality with a reasonable privacy loss.

The second challenge is to preserve the structure of the data in private data synthesis. 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. As a result, the synthesized data is not useful for any downstream tasks that require this structure to be preserved. We propose KAMINO, a data synthesis system to ensure differential privacy and to preserve the structure and correlations present in the original dataset. 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.

The third challenge is efficient and secure private data profiling. Discovering functional dependencies (FDs) usually requires access to all data partitions to find constraints that hold on the whole dataset. Simply applying general secure multi-party computation protocols incurs high computation and communication cost. We propose SMFD to formulate the FD discovery problem in the secure multi-party scenario, and design secure and efficient cryptographic protocols to discover FDs over distributed partitions. Experimental results show that SMFD is practically efficient over non-secure distributed FD discovery, and can significantly outperform general purpose multi-party computation frameworks.

Acknowledgements

I would like to express my most sincere gratitude to my PhD advisor Prof. Ihab Ilyas. He has always been my inspiration and my role model as a researcher, a professor, an entrepreneur, and a human being. Throughout the years, he pulled me back every time when I was chasing butterfiles, and offered invaluable training in both professionalism and mentalism, which I will carry and treasure forever.

I would like to extend my sincere thanks to my advisory committee members Prof. Xi He, and Prof. Florian Kerschbaum for their support, discussions, and their assistance at every stage of my PhD journey. I would like to offer my special thanks to Prof. Amol Deshpande for serving as the External Examiner, and to Prof. Helen Chen for serving as the Internal-External Member of my examining committee. I am deeply grateful to them for spending time reading and providing valuable comments on my thesis. Without my examining committee members, this dissertation could not have been possible.

I am extremely grateful and proud to have the best supportive family. I wouldn't have had the opportunities to pursue and focus on PhD without their endless love, support, patience and trust. All words are too weak to express my feelings, and I am forever indebted to them.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data analysis is a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making [34]. Data analytics is being widely used not only as a business tool, which empowers organizations to drive efficiencies, glean deeper operational insights and identify new opportunities, but also for the greater good of society, as it is helping solve some of world's most pressing issues, such as developing COVID-19 vaccines, fighting poverty, inequality, and climate change.

In a typical setting for data analytics, the data scientist accesses the data and conducts analytics tasks, such as issuing queries in structured query language (SQL), connecting Tableau [130] to draw interactive visualizations, training neural networks on TensorFlow [10] to forecast sales and etc., as long as the data scientist has the permission to access the data. However, the prerequisite of having direct data access does not always hold true when analyzing data in enterprises. Specifically, conducting enterprise data analytics faces two practical problems. First, many datasets contain sensitive information about persons and business secrets, and hence those data have restricted access and do not allow unfettered access; Second, there are multiple business units within an organization, and each of them owns a part of the overall data assets. Because of independence of each business unit as well as privacy legislations such as GDPR [7], each of the data are owned and stored in silos by multiple parties, with different access control. Therefore, data scientists have to do analytics on private and siloed data.

1.1 Data Analytics on Private Data Silos

For accessing private data silos, the current practices adopted by enterprises include having the data scientist go through privacy clearance and approvals, sign non-disclosure agreement, which are lengthy and not scalable. Furthermore, those practices do not provide any technical guarantees against privacy breaches, which often lead to disastrous consequences in financial loss [1, 2, 4], reputational damage [3, 89], and most importantly, loss of sensitive data [6, 18, 87].

The problem of data analytics on private siloed data is fundamentally challenging due to competing goals. On the one side, achieving good quality data analytics requires understanding the whole picture of the data. To the data scientist's interest, the data scientist wants to extract useful insights as many as possible. This is done through conducting a diverse set of tasks to understand the data, from simply eyeballing the data to many more dedicated tasks, such as exploring data [83], profiling metadata [94], and training machine learning models [28]. In addition, it is often necessary to combine information from multiple data silos to form a whole picture of the underlying data [96]. However, most data scientists are not data privacy and security experts and hence they do not have the technical capability to defend privacy and security in their jobs. On the other side, data privacy and security requirements mandate least information leakage. To the interest of data owner, the data owner wants to prevent data leakage, and ideally, nothing should be revealed.

Therefore, in order to balance the interest from both sides, data analytics tasks on private data silos have to make tradeoffs between analytics utility (e.g., statistics, dependencies and trends) and data privacy (e.g., deniability and indistinguishability). However, it is difficult to implement such a tradeoff in practice, because most existing solutions do not serve both interests, naturally. For example, in the past decades, many privacy-preserving techniques have been designed, such as differential privacy [51, 56], secure computation including homomorphic encryption [110], oblivious polynomial evaluation [131] and many other remarkable schemes [25, 31, 45]. Those techniques ensure certain level of privacy and security to the data owner, but they misalign with the practical aspects that data scientist cares about, such as accuracy, performance and other utility goals.

1.2 Scope and Challenges

To enable data analytics on private siloed data, we identify two common interactions around sensitive data shown in Figure 1.1. The first interaction is between the data scientist

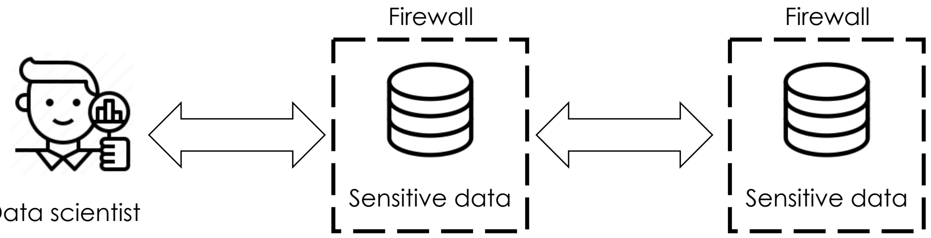


Figure 1.1: Two interactions over the the sensitive dataset: 1) between the data scientist and the sensitive data; 2) among multiple sensitive datasets.

and one sensitive dataset, and the other interaction is among multiple sensitive datasets federated together to complete the task curated by the data scientist.

1.2.1 Data Scientist Interacting with Private Data

For the data scientist interacting with private dataset, we study two common approaches. The first approach is to bring the data scientist close to the sensitive data, and we enable **data exploration** as an instance of this approach, which allows the data scientist to interactively issue SQL queries to explore the sensitive data. While there exist general purpose differentially private query answering systems, they are not really meant to support interactive querying, and they fall short in two key respects. First, these systems place an inordinate burden on the data scientist to understand differential privacy and differentially private algorithms. For instance, PINQ [126] and wPINQ [145] allow users to write differentially private programs and ensure that every program expressed satisfies differential privacy. However, to achieve high accuracy, the data scientist has to be familiar with the privacy literature to understand how the system adds noise and to identify if the desired accuracy can be achieved in the first place. ϵktelo [173] has high level operators that can be composed to create accurate differentially private programs to answer counting queries. However, the data scientist still needs to know how to optimally apportion privacy budgets across different operators. FLEX [93] allows users to answer one SQL query under differential privacy, but has the same issue of apportioning privacy budget across a sequence of queries. Second, and somewhat ironically, these systems do not provide any guarantees to the data scientist on the quality they really care about, namely *accuracy* of query answers. In fact, most of these systems take a privacy level as input and make sure that differential privacy holds, but leave accuracy unconstrained. Therefore, we aim to design a system that allows data scientist to explore a sensitive dataset held by a data owner by posing a sequence of declaratively specified queries that can capture typical data exploration workflows. The system aims at achieving the following dual goals: 1) since the data are sensitive, the data owner would like the system to provably bound the information disclosed about any one record to the data scientist; and 2) since privacy preserving mechanisms introduce error, the data scientist must be able to specify an accuracy bound on each query.

The second approach for the data scientist interacting with private dataset is from the other direction, by generating a similar instance and bringing it to the data scientist on the outside of the firewall. This is usually done through the task of **data synthesis**, which samples a synthetic database instance and releases it to the data scientist for running downstream applications. 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 [103] or key constraints [59] between attributes and tuples. 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. For example, to answer statistical queries, prior work [30, 55, 65, 159] have shown that the running time for generating 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., [20, 38, 39, 95, 174]) 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. As a result, the structure of data in the output database instance can be significant different from that in the original data. 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.

1.2.2 Interaction among Multiple Private Data

The second type of interaction shown in Figure 1.1 is among multiple parties, where each of them holds a data partition and is federated to complete one task curated by the data scientist. We would like to protect the input data of any party from all other parties, and this thesis enables the task of **data profiling** by securely profiling functional dependencies (FDs) over the union of all private data partitions. Given these partitioning requirements on the data, the question is how to efficiently discover FDs that hold on the whole dataset, while minimizing the data leakage in the lack of a single trusted party. In fact, the question presents a 3-way tradeoff between *soundness*, *privacy* and *efficiency*: soundness requires exchanging information among partitions to correctly compute the global set of FDs; privacy dictates not revealing more than necessary data to other partitions and without revealing certain information under some acceptable definition of privacy; and efficiency necessitates the overall process to run in reasonable (e.g., polynomial) time. Naïvly applying current techniques can be either insecure, incorrect or too expensive. For example, simply computing FDs from each partition separately and intersecting the FD sets might respect the privacy of each partition, but can lead to invalid FDs on the whole dataset. On the other hand, exchanging information between partitions to validate global FDs in an encrypted way might meet both privacy and soundness, but suffers from severe inefficiency. For instance, validating one candidate FD on a dataset requires comparing all needed evidence from all the partitions. Securely comparing all tuples from all partitions is expensive due to the lack of scalable cryptographic constructions. General purpose secure multi-party computation protocols such as garbled circuit [169], homomorphic encryption [110] and oblivious polynomial evaluation [131] incur high cost already when securely computing one function among multiple parties [119], rendering an FD discovery algorithm prohibitively inefficient. Therefore, it would be desirable to design efficient and secure protocols for discovering FDs.

1.3 Contributions and Outline

To solve the aforementioned challenges in enabling private data analytics, we first study each task and express its utility as data constraints (e.g., integrity constraints [59, 88, 103]). Then, we select a principled data privacy and security model for each task, such as input privacy [72] and output privacy [51, 56]. Next, we design mechanisms to combine them into end to end analytics tasks. Finally, we develop open source systems to support the evaluation and deployment of each task. The contributions of this dissertation are summarized as follows.

1.3.1 Data Exploration

We propose $\mathcal{A}$PEx [68], an accuracy-aware privacy engine for sensitive data exploration (§ 3). The data scientist can interact with the private data through $\mathcal{A}$PEx using declaratively specified aggregate queries. Specifically, we make the following contributions:

1. $\mathcal{A}$PEx supports three types of aggregate queries (§ 3.1.1): 1) *workload counting queries* that capture the large class of linear counting queries (e.g., histograms and CDFs), which are a staple of statistical analysis; 2) *iceberg queries*, which capture HAVING queries in SQL and frequent pattern queries; and 3) *top-k queries*. These queries form the building blocks of several data exploration workflows. To demonstrate their applicability in real scenarios, we express two important data cleaning tasks, namely blocking and pair-wise matching, using sequences of queries from our language (§ 3.6).

2. In our language, each query is associated with intuitive accuracy bounds that permit $\mathcal{A}$PEx to use differentially private mechanisms that introduce noise while meeting the accuracy bound (§ 3.1.2).

3. For each query in a sequence, $\mathcal{A}$PEx employs an *accuracy translator* (§ 3.3) that finds a privacy level and a differentially private mechanism that answers the query while meeting the specified accuracy bound. For the same privacy level, the mechanism that answers a query with the least error depends on the query and dataset. Hence, $\mathcal{A}$PEx implements a suite of differentially private mechanisms for each query type, and given an accuracy bound chooses the mechanism that incurs the least privacy loss based on the input query and dataset.

4. $\mathcal{A}$PEx uses a *privacy analyzer* (§ 3.4) to decide whether to answer a query such that the privacy loss to the data owner is always bounded by a budget. The privacy analysis is novel since 1) the privacy loss of each mechanism is chosen based on the query's accuracy requirement; and 2) some mechanisms have a data dependent privacy loss.

In a comprehensive empirical evaluation on real datasets with query and application benchmarks, we demonstrate that 1) $\mathcal{A}$PEx chooses a differentially private mechanism with the least privacy loss that answers an input query under a specified accuracy bound; and 2) allows data scientists to accurately explore data while ensuring provable guarantee of privacy to data owners (§ 3.5).

1.3.2 Data Synthesis

We consider an important class of structure constraints, the *denial constraints* (DCs) [88], and we present KAMINO [70], a system for constraint-aware differentially private data synthesis (§ 4).

Our solution is built on top of the probabilistic database framework [151, 156], 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 [165] and the attention mechanism [19] 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:

1. 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.

2. 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 (§ 4.1 & § 4.2).

3. 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 (§ 4.3).

4. 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 (§ 4.4).

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 (§ 4.5).

1.3.3 Data Profiling

To solve the problem of discovering global FDs among multiple parties securely and efficiently, we propose SMFD [69], a system with efficient cryptographic protocols to support discovering FDs in semi-honest multi-party scenarios (§ 5). We highlight the main contributions of SMFD as follows:

1. We define the FD discovery problem in the secure multi-party scenario (§ 5.1.1) and propose a top-down based framework to validate FDs (§ 5.1.3).

2. We formulate the distributed FD validation problem over multiple partitions (§ 5.2.1), and provide an efficient solution for discovering FDs in the multi-party scenario (§ 5.2.2).

3. As the building blocks of our solution, we design efficient mix networks to enable secure equality testing against a semi-honest adversary (§ 5.3).

4. We also propose a relaxed version of FD (referred to as congenial FD) and show that our framework is able to efficiently and securely discover these FDs (§ 5.4).

With extensive evaluations using real world datasets, we show the linear scalability of our solution, and empirically demonstrate that our solutions can achieve more than two orders of magnitude improvement over general purpose secure multi-party computation solution, in terms of computation and communication costs (§ 5.5).

1.3.4 Other Tasks

This dissertation addresses the specific technical challenges in the three analytics tasks introduced above, and many other challenges remain unsolved in the diverse spectrum of data analytics tasks under the scope described by Figure 1.1. We will discuss the challenges in other data analytics tasks and present directions for future research (§ 6). □

The remainder of the dissertation is organized as follows. In § 2, we start with background and related work. In § 3, we present our approach for accuracy-aware differentially private data exploration. In § 4, we introduce our approach for constraint-aware differentially private data synthesis; and in § 5, we present our approach for secure multi-party functional dependency discovery. Finally, in § 6, we discuss future directions and conclude this dissertation.

Chapter 2

Preliminaries and Related Work

2.1 Preliminaries

2.1.1 Relational Data and Constraints

We consider the sensitive dataset in the form of a single-table relational schema $R(A_1, A_2, \ldots, A_d)$ with d attributes, where $attr(R)$ denotes the set of attributes of R. Each attribute A_i has a domain $dom(A_i)$. The full domain of R is $dom(R) = dom(A_1) \times \cdots \times dom(A_d)$, containing all possible tuples conforming to R. An instance D of relation R is a multiset whose elements are tuples in $dom(R)$. 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. We let the domain of the instances be $\mathcal{D}$ and use index 1 to refer to the first element in an array.

Functional Dependency

Definition 1. *(Functional Dependency) Given a data instance D in schema R, let A be a set of attributes $A \subseteq R$ and B be an attribute $B \in R$, a functional dependency (FD) $f : A \to B$ holds on D (i.e., $D \models f$) if for all tuples $t_1, t_2 \in D, t_1[A] = t_2[A] \Rightarrow t_1[B] = t_2[B]$.*

Note that we follow the literature to consider only the *regular* FDs where the B contains one single attribute. Any irregular FD is equivalent to a set of regular FDs. When $B \in A$, $A \to B$ is *trivial*. An FD is *minimal* if $\nexists K \subseteq A$ such that $A \setminus K \to B$ is a valid FD. For $K \subseteq R \setminus A$, $f' : A, K \to B$ is called a specialization of f, i.e., $f' \in Spec(f)$.

Example 1: Consider the following FDs:

$f_1 : edu \rightarrow edu\_num$

$f_2 : edu, edu\_num \rightarrow edu\_num$

$f_3 : edu, age \rightarrow edu\_num$

f_1 states that for any two persons with the same *edu*, they must have the same *edu_num*. Given a table D shown by Figure 2.1, it is easy to validate that $D \models f_1$.

	age	edu	edu_num
t_1	35	13	Bachelor
t_2	42	13	Bachelor
t_3	27	14	Master
t_4	55	14	Master
t_5	67	14	Master

Figure 2.1: A simple table example.

FD f_2 is trivial, and FD f_3 is not minimal because of f_1. □

Definition 2. *(Attribute Partition) Given an attribute set A and a relational instance D, the attribute partition of D under A is a disjoint set as $\pi_A^D = \{[t]_A | t \in D\}$, where $[t]_A = \{t' \in D | t[A'] = t'[A'], \forall A' \in A\}$*

$|\pi_A^D|$ represents the cardinality of the set π_A^D. Since π_A^D is a set of tuple sets, we use $\|\pi_A^D\|$ to denote the cardinality of tuples, which in other words, is equivalent to the cardinality of D. We use V_A^D to represent the set of values of the attribute partition of A on D.

Example 2: Continue with Example 1, The attribute partition on *edu* is $\pi_{edu} = \{\{t_1, t_2\}, \{t_3, t_4, t_5\}\}$, and thus $|\pi_{edu}| = 2$. □

Definition 3. *(Attribute Partition Error) The attribute partition error $e(A)$ of an attribute set A with respect to an instance D is defined as the minimal number of rows that need to be removed in order to make A a super key. Mathematically, $e(A^D) = \|\pi_A^D\| - |\pi_A^D|$.*

Example 3: Continue with Example 2, The attribute partition error on *edu* is $e(edu^D) = 5 - 2 = 3$, which implies that we need to remove at least 3 rows in order to make attribute *edu* as a super key on table D showed by Figure 2.1. □

Denial Constraints

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 $v_1, v_2 \in t_x[A]$, $x \in \{i, j, \cdots\}$, $A \in R$, $o \in \{=, \neq, >, \geq, <, \leq\}$, and c is a constant. We will omit universal quantifiers $\forall t_i, t_j, \ldots$ hereafter for simplicity.

Example 4: 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 the FD f_1 from Example 1. 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, $\mathcal{A}_{\phi_1} = \{edu, edu\_num\}$.

2.1.2 Probabilistic Database

Probabilistic Database Modeling

The probabilistic database framework [151] has been used in practice [148, 165] 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:

$$\Pr(D) \propto \prod_{t \in D} \Pr(t) \times \exp(-\sum_{\phi \in \Phi} w_\phi \times |V(\phi, D)|) \tag{2.1}$$

where $\prod_{t \in D} \Pr(t)$ models a tuple-independent probabilistic database [151, 156], 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.

Probabilistic Database Learning

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 [151]. The observed data will be used to learn the parameters in the model. We assume the distribution does not change.

The tuple probability can be expressed as the product of a chain of conditional probabilities:

$$\Pr(t) = \Pr(t[A_1]) \prod_{j=2}^{k} \Pr(t[A_j] \mid t[A_1, \cdots, A_{j-1}]) \tag{2.2}$$

Each conditional probability is learned as a discriminative model based on *tuple embedding* [165] and *attention mechanism* [19]. Similar to word embedding that models words in vectors of real numbers [127], tuple embedding has been applied to model tuples by encoding tuples into the space of real numbers [57, 165].

Consider the discriminative model used in AimNet [165] 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 $\vec{x} \in \mathbb{R}^{d'}$, 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: $\vec{z} = B\omega(A\underline{\vec{x}} + \vec{c}) + \vec{d}$, where A, B, $\vec{c}$, $\vec{d}$ are learned parameters and ω is a ReLU. For each discrete attribute, AimNet associates it with a learnable lookup table mapping embeddings to domain values.

AimNet relies on the attention mechanism [19] 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.

2.1.3 Differential Privacy

Differential privacy (DP) [51, 56] aims to protect the output of has emerged as a standard data privacy guarantee by government agencies [13, 80] and companies [60, 74, 93]. An algorithm that takes as input a table D satisfies differential privacy if its output does not significantly change by adding or removing a single tuple in its input.

Definition 4 (Differential Privacy (DP) [51, 56])**.** *A randomized mechanism $\mathcal{M}$ 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). When $\delta = 0$, we say the randomized mechanism $\mathcal{M}$ is ϵ-DP. Smaller values of ϵ result in stronger privacy guarantees as D and D' are harder to distinguish using the output. Complex DP algorithms can be built from the basic algorithms following two important properties of differential privacy.

The sequential composition theorem helps assess the privacy loss of multiple differentialy private mechanisms.

Theorem 1 (Sequential Composition [51])**.** *Let $M_1(\cdot)$ and $M_2(\cdot, \cdot)$ be algorithms with independent sources of randomness that ensure (ϵ_1, δ_1)- and (ϵ_2, δ_2)-differential privacy respectively. An algorithm that outputs both $M_1(D) = O_1$ and $M_2(O_1, D) = O_2$ ensures $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-differential privacy.*

Another important property of differential privacy is that post-processing the outputs does not degrade privacy.

Theorem 2 (Post-Processing [52])**.** *Let $M_1(\cdot)$ be an algorithm that satisfies (ϵ, δ)-differential privacy. If applying an algorithm M_2 the output of $M_1(\cdot)$, then the overall mechanism $M_2 \circ M_1(\cdot)$ also satisfies (ϵ, δ)-differential privacy.*

All steps in the post-processing algorithm do not access the raw data, so they do not affect the privacy analysis.

Semantics of DP. Prior work [21, 54, 67, 82, 98, 99, 147] 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. [67] formalize the notion of "external knowledge", and of "drawing conclusions" respectively via (i) a *prior* probability distribution $b[\cdot]$ on the database domain $\mathcal{D}$ of size n; and (ii) the corresponding *posterior* probability distribution: given a transcript t outputted by mechanism $\mathcal{M}$, the adversary updates his belief about the database D using Baye's rule to obtain a posterior $\hat{b}[D|t] = \frac{\Pr[\mathcal{M}(D)=t]b[D]}{\sum_{D'} \Pr[\mathcal{M}(D')=t]b[D']}$. Consider the hypothetical scenario where person i's data is not used, denoted by $\mathcal{M}(D_{-i})$, given a transcript t, the updated posterior belief about the database D is defined as $\hat{b}_i[D|t] = \frac{\Pr[\mathcal{M}(D_{-i})=t]b[D]}{\sum_{D'} \Pr[\mathcal{M}(D'_{-i})=t]b[D']}$. A DP mechanism $\mathcal{M}$ with proper privacy parameters (sufficiently small δ) can prevent the adversary from drawing different conclusions about whether or not person i's data was used, i.e., the statistical difference between $\hat{b}[\cdot|t]$ and $\hat{b}_i[\cdot|t]$ is small for D drawn from $b[\cdot]$ and t drawn from $\mathcal{M}(D)$ with a high probability [67, 98]. This posterior-to-posterior comparison applies to arbitrary prior of adversaries, unlike prior-to-posterior approaches [54, 99].

2.1.4 Cryptographic Constructions

ElGamal Encryption

ElGamal encryption [66] is a probabilistic asymmetric encryption scheme. ElGamal can be defined over a cyclic group $\mathbb{G}_p$ of order p with a generator g. p is usually a large prime number and g is a primitive element of the group. ElGamal encryption consists of three algorithms: key generation, encryption and decryption, which are formally described in Algorithm 1.

The KGen algorithm takes two input values: a large prime number p, and the generator g. The private key x is a random integer uniformly drawn from $\mathbb{Z}_p$, and the public key k is an element in $\mathbb{G}_p$ computed from x and g. By the discrete logarithm assumption [50], it is difficult to infer x from k.

The Enc algorithm takes inputs of a value v and the public key k, and outputs a cipher text pair (c_1, c_2). The Dec algorithm decrypts the cipher text pair using private key x.

Algorithm 1 ElGamal Scheme

```
Input: $p, g$                                        ▷ a large prime, generator
    procedure KGen($p, g$)
        $x \leftarrow_{\$} \mathbb{Z}_p$                               ▷ $x$ is the private key
        $k \leftarrow g^x$                                 ▷ $k$ is the public key
        return $(x, k)$
    end procedure
```

```
Input: $v, k$                                        ▷ plain value, public key
    procedure Enc($v, k$)
        $r \leftarrow_{\$} \mathbb{Z}_p$                               ▷ sample a random value
        $c_1 \leftarrow g^r$
        $c_2 \leftarrow v \cdot k^r$
        return $(c_1, c_2)$
    end procedure
```

```
Input: $(c_1, c_2), x$                               ▷ cipher text, private key
    procedure Dec($(c_1, c_2), x$)
        $c_1' \leftarrow c_1^x$                            ▷ updating random parameter
        $c_2 \leftarrow c_2 \cdot (c_1')^{-1}$                     ▷ decrypt
        return $c_2$
    end procedure
```

Mix Network

A mix network (mixnet, in short) [36] is a cryptographic construct where a chain of *workers* establishes hard-to-trace communications between the senders and receivers. The senders encrypt each message to each worker using public key cryptography and send encrypted messages to the mixnet, where each worker strips off an encryption layer, does some operations and shuffles them before enrouting to the next worker sequentially. At the output, receivers receive cryptographically transformed and randomly permuted messages, making the end-to-end communications untraceble.

2.2 Related Work on Specific Private Data Analysis Tasks

2.2.1 Private Data Exploration

Data exploration refers to a process in data analysis, where a data scientist uses visual exploration to understand what is in a dataset and the characteristics of the data. These characteristics can include size, quantity, and quality aspects such as completeness and correctness of the data, which are the output results from the exploration step. Therefore, the input parties would like to have output privacy guarantee for the private data exploration tasks. In this section, we review the approach of private data exploration through query answering.

As discussed in § 2.1.3, differential privacy [53] has emerged as a popular output privacy standard in real-world analysis, and has been widely adopted by both the government agencies [13, 35] and enterprise [60, 74, 78, 92, 122]. In the past decade, differentially private query answering has been extensively studied, and there are well known techniques for special tasks such as answering linear counting queries [81, 114, 123], top-k queries [111] and frequent itemset [116, 172]. Due to the diversity of query types, there is no single technique that fits all, and hence existing work only study a subset of query types and design specific mechanisms to answer those queries. For example, PrivateSQL [106] proposes methods to answer join queries over multiple tables by truncating sensitivity from the query plans. HDMM [124] designs a compact implicit matrix representation and exploits this representation to efficiently optimize over (a subset of) the space of differentially private algorithms with low expected error. As a result, for a given query, there is no clear winner between these proposed algorithms.

At a lower level, there are programming frameworks such as PINQ [126], wPINQ [145] and ϵktelo [173] to allow users to write programs in higher level languages for various tasks. These systems automatically prove that every program written in this framework ensures differential privacy. For example, FLEX [93] allows users to answer a SQL query under a specified privacy budget. However, using those general purpose system poses technical barriers to the data scientist, as they require the data scientist to specify a privacy level, and more importantly, to understand how a query is answered. For example, In ϵktelo [173], a differentially private program is described as a plan over a high level library of operators. Operators are organized into classes based on their functionality, which includes data transformations, data reductions, query selection and execution, and inference methods to combine noisy answers into a consistent estimate. Understanding and writing those programs are sometimes hard for data scientists, who are in general not privacy experts.

Despite the fact that designing new mechanisms to preserve the output privacy for more query types is an active research, there is significantly less work on addressing accuracy in differentially private query answering. Since differential privacy is achieved by randomization and noise is added to the output (or, to the intermediate steps by the post-processing property by Theorem 2), the noisy outputs do not enjoy the same level of accuracy compared to those without randomization. To the perspective of data scientists, query accuracy is critical to achieve correctness. In fact, the lose of accuracy has been a bottleneck to adopt differential privacy in some cases and prioritizing accuracy for diverse use cases has been a difficult task even for experts [80]. However, those systems mentioned above only take a privacy level and ensure the differentially private answers for query answering, but leaving query accuracy unconstrained.

Accuracy constraints in DP. We review the related work of addressing the accuracy constraints in differentially private query answering.

At the time of designing our accuracy-aware differentially private data exploration engine (APEx in § 3), the most related work to support accuracy constraints is by Ligett et al. [117], which allows the data scientist to specify accuracy constraints. Ligett et al. takes as input a *given mechanism with accuracy bound and a set of privacy budgets*, $(M, \alpha, \beta, \{\epsilon_1 < \epsilon_2 < \cdots < \epsilon_T\})$, and outputs the minimal privacy cost (epsilon) for M to achieve the accuracy bound. Rather than concentrating on private machine learning theoretically, our focus is to explore private data with accuracy guarantees. The main technical differences are highlighted below.

First, the end-to-end problems in APEx and the approach in Ligett et al. [117] are different. APEx aims to translate a *given query with accuracy bound*, (q, α, β), to a differentially private mechanism that achieves this accuracy bound with the minimal privacy cost, (M, ϵ). On the other hand, Ligett et al. [117] takes as input a *given mechanism with accuracy bound and a set of privacy budgets*, $(M, \alpha, \beta, \{\epsilon_1 < \epsilon_2 < \cdots < \epsilon_T\})$, and outputs the minimal privacy cost (epsilon) for M to achieve the accuracy bound. Unlike Ligett et al., APEx does not need to take a set of privacy budgets as an input. For the exploration queries in APEx, there are more than just one differentially private mechanisms for each query type, and none of these mechanisms dominate the others. Thus, in this sense, the problem solved by APEx is more general than the one solved by Ligett et al.

The second key difference between the approaches is the following. APEx currently only supports mechanisms for which the relationship between the accuracy bound and the privacy loss epsilon can be established analytically. On the other hand, Ligett et al. can handle arbitrarily complex mechanisms, and use empirical error of the mechanisms to pick the epsilon. In this way, the solution proposed by Ligett et al. applies a larger class of mechanisms. Nevertheless, for the queries and mechanisms supported by APEx, using Ligett et al.'s methods would be overkill in two ways: 1) there is an extra privacy cost ϵ_0 to test the empirical error; and 2) since the exploration queries are variations of counts, the sensitivity of the error will be so high that the noise introduced to the empirical error could limit our ability to distinguish between different epsilon values. For example, for a simple counting query of size 1, the maximum privacy cost required in APEx is $\ln(1/\beta)/\alpha$ while the privacy cost with Ligett et al. is more than $\epsilon_0 = 16(\ln(2T/\beta))/\alpha$. When APEx applies a data-dependent approach, the privacy cost can be even smaller than $\ln(1/\beta)/\alpha$. For a prefix counting query of size L and sensitivity of L, the best privacy cost achieved in APEx is $O(\log L)$, but their ϵ_0 is $O(L)$ as the sensitivity of the error to the final query answer is L (for both Laplace and strategy-based mechanisms). Similarly, for iceberg counting queries (ICQ), the sensitivity of error to the final query answer is large and results in a large ϵ_0 for the differentially private testing. Thus, using the method in Ligett et al. would not help the currently supported mechanisms and queries in APEx. In the future, we will add more complex mechanisms into APEx (like DAWA [112], MWEM [79] that add data dependent noise) and study whether the methods of Ligett et al. can be adapted to our setting.

APEx has attracted many attentions and inspired follow-up works [102, 161] on the line of explicitly ensuring accuracy guarantee in differentially private query answering. DPella [161] is a programming framework providing data scientists with support for reasoning about privacy, accuracy, and their trade-offs. DPella statically tracks the accuracy of different data analysis, and provides tight accuracy estimations using taint analysis. Based on APEx, a recent thesis proposal [102] presents a differentially private framework that handles multiple data scientists with their individual accuracy guarantees when given a limited privacy budget.

Private QA for data cleaning. Differentially private query answering provides a way for the data scientist to explore the data, and can be used to help tune a cleaning workflow on the private data. Amongst prior work on cleaning private data [41, 77, 85, 109], the most relevant ones are PrivateClean [109] and HRR [77] as they both use differential privacy too. However, similarities to APEx stop with that. For PrivateClean [109], it assumes a different setting, where no active data cleaner is involved. The data owner perturbs the dirty data without cleaning it, while the data analyst who wants to obtain some aggregate queries will clean the perturbed dirty data. Moreover, all the privacy perturbation techniques in PrivateClean are based on record-level perturbation, which 1) only work well for attributes with small domain sizes; and 2) has asymptotically poorer utility for aggregated queries. At last, the randomized response technique used for sampling string attributes in PrivateClean does not satisfy differential privacy – it leaks the active domain of the attribute. As for

HRR [77], it deals with multi input party settings, where each input party perturb data using the local model of differential privacy, and then the computation node applies rules such as functional dependency (§ 2.1.1) to rectify violations.

2.2.2 Private Data Sampling

There has been extensive literature on releasing synthetic relational data with differential privacy guarantee [33, 62, 70, 133, 177]. In general, generating differentially private synthetic data is hard, due to the tradeoff between accuracy and privacy [30, 55, 65, 159]. 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., [20, 38, 39, 95, 174]) 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. The main approach for existing method is to privately learn a data generative model, and then sample independent and identically distributed tuples as the post-processing step. Depending on how to model the data generative process, prior approaches can be categorized into two main classes: 1) statistical approaches, which focus on exploiting certain statistical property of the dataset; and 2) deep learning approaches, which train a deep generative model to sample tuples.

Statistical approaches. Statistical approaches produce synthetic data by exploiting the statistical property of the dataset. We classify two subcategories of methods based on the the statistical properties they utilize.

One popular subcategory of statistical approaches is through *dimensionality reduction*. Due to the hardness of privatizing high-dimensional data with differential privacy guarantee [30, 55, 65, 159], the distribution of the tuples can be estimated by low-dimensional marginal distributions [146, 167] based on the assumption of conditional independence among attributes. Those low-dimensional distributions are easier to add noise due to the relatively lower noise-to-signal ratio [174]. Then, the tuple distribution can be modeled using probabilistic graphical models [105], such as using the directed network [115, 144, 174] or undirected graphs [39, 125]. For example, given a dataset, PrivBayes [174] first constructs a Bayesian network, which 1) provides a model of correlations among attributes; and 2) allows to approximate the tuple distribution. After that, it injects noise into each marginal to ensure differential privacy, and then uses the noisy marginals and the Bayesian network to construct an approximation of the data distribution. Finally, it samples tuples from the approximate distribution to construct a synthetic dataset, and then releases the synthetic data. To improve the scalability and generality, PGM [125] proposes to use an undirected graph to do proper inference over private observations such as resolving inconsistency from noisy distributions. Nevertheless, under those dimensionality reduction models, only correlations among dependent attributes are likely to be captured, but correlations that widely exist among conditional independent attributes and more importantly, tuples such as constraints (§ 2.1.1) are not captured in prior work.

Another subcategory of statistical approaches is the *game-based* methods [65, 79, 162], where a set of workload is taken as additional input, and the goal is to sample private synthetic data, on which the workload performs well. In the game-based methods, producing differentially private data is viewed as a zero-sum game [84] between a data player and a query player. The players adopt an adversarial learning framework (e.g, [73]) to gradually improve the quality of the synthetic dataset. For example, MWEM [79] can be viewed as a game, where the data player maintains the distribution over the data domain using multiplicative weights [17], and the query player selects a query on which the maintained distribution has the worst performance, with the goal of separating the synthetic dataset from the real one. For another example, Dual Query [65] considers a dual formulation of the MWEM problem, where the query player runs multiplicative weights over the input workload, and the data player generates a tuple that minimizes the error on selected queries, which can be computed via integer programming. The game-based methods adopt learning-theoretical frameworks [30, 97], and often provides nearly optimal theoretical guarantees on workload accuracy. However, in those methods, optimizing for the specific set of queries increases the chance of over-fitting, and may not generalize in the presence of dynamic workload.

Statistical methods for private data sampling rely on the certain statistical property of the dataset, and are often easy to model, interpret and inference. The cons of those methods usually come from the simplification of modeling real-world relational data due to their limited representation and learning power. For example, non-linear correlations between attributes are often not captured by statistical methods [91]. Furthermore, all above methods only model the correlations between attributes, and have an implicit assumptions that tuples are independent and identically distributed, which is generally not true in relational data, where correlations among tuples do exist (§ 2.1.1).

Deep learning approaches. Deep learning models have been widely used in synthesizing unstructured data, such as images [154], videos [37] and natural languages [76]. Unlike statistical approaches, deep learning approaches for private data sampling can capture the complex (non-linear) correlations among attributes. In this section, we review two subcategories of deep learning approaches based on the model they use.

The first category uses the generative adversarial network (GAN [73]), where a generator and discriminator pit one against the other [64, 157, 168, 175]. There is a rich class of GAN variants that have been used in private data sampling (see the survey [62]), and in section, we briefly review two representative class based on the modeling training methods with differential privacy. The training is an iterative process, and differential privacy can be achieved using DP-SGD [11, 22, 155, 164]. The trained generator serves as the generative model, and sampling private data is a post-processing step without incurring more privacy loss (Theorem 2). For example, DPGAN [168] and dp-GAN [175] are two independent methods proposed to learn differentially private GANs. Both DPGAN and dp-GAN adopt the idea of DP-SGD framework to train the models privately, but they slightly differ in their object functions. To be more precise, DPGAN adopts the Wasserstein GAN objective [16], while dp-GAN adopts the improved Wasserstein GAN [75], which is an alternative to weight clipping in order to enforce the Lipschitz constraint.

In addition to the private model training using DP-SGD, private model training can also leverage the ensembles [139, 140]. In the PATE framework [140], multiple teachers are trained on disjoint sensitive data (e.g., different users' data), and uses the teachers' aggregate consensus answers in a black-box fashion to supervise the training of a student model. By publishing only the student model (keeping the teachers private) and by adding carefully-calibrated noise to the aggregate answers used to train the student, the PATE framework can achieve differential privacy. PATE-GAN [95] adopts the PATE framework to achieve differential privacy, and proposes a method to train the student discriminator without requiring publicly available dataset. Specifically, a set of teacher discriminators are trained separately as the standard teacher models on disjoint partitions of the training set. A student discriminator is trained with generated samples, labeled by the teachers using the PATE framework. The generator is trained to minimize its loss with respect to the student discriminator. As a result, the student model can be trained privately without public data and the generator can utilize the process to improve the generated samples.

The second category of deep-learning based private data sampling uses auto-encoder [100], which learns a representation for the private input tuples by training the network, with differential privacy guarantee [38, 143, 149]. The privately learned representation can be used to sample tuples first in the latent space, and then convert back to the database domain. Similar to the methods using GANs, differential privacy can be achieved by perturbing the gradients using DP-SGD during each iteration of training, or perturbing the objective function by injecting noise. For example, DP-VaeGM [38] is trained on the private data by perturbing the gradients and is then released to public for generating synthetic data. As an example of perturbing the objective function, recent work [149] proposes a functional perturbation mechanism, where the global sensitivity of the loss function is carefully derived. In each iteration, the difference of input data only influences the coefficients of the model parameters and it only needs to compute the sensitivity of the coefficients and add perturbation to them. After gaining the sensitivity of objective function, they add perturbation to every coefficients in the loss function. Finally, they use it to train and adjust the parameters of the model by minimizing the loss function. □

Different from unstructured data, structured data is defined using relational schema and hence, structure correlations naturally exist, as we have introduced in § 2.1.1. Given the correlations and dependencies among tuples and attributes, applying deep learning models on structured data faces at least two issues. 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 [61], Second, similar to statistical approaches, methods based on deep models (e.g. [64, 95, 157, 168]) suffer from missing structure correlations.

Utility goals in private data synthesis. The utility goals in existing differentially private data synthesis can be classified into two classes. The first class are *data-independent* goals, such as k-way marginal distances [158] used by most of the statistical approaches, misclassification rate [29] by many of the deep learning approaches. The second class are *data-dependent* goals, such as query error [65] in the game-based approaches discussed above.

Our approach in Chapter 4 distinguishes from prior work in three aspects. First, it is a combination, using both the statistical and deep learning methods to model the distribution of databases; Second, our method differentiates prior work in that we explicitly consider the denial constraints [88] enforced among tuples as an utility goal; Last but not least, our approach lifts the assumption of i.i.d tuples.

2.2.3 Private FD Profiling

Discovering functional dependencies (FDs) has been studied for a long time, and there are a few surveys [88, 120, 136, 137] to summarize those discovery methods. Prior work on functional dependency discovery can be categorized into four classes.

The first class includes top-down schema driven approaches, which generate candidate FDs first using the schema, and then remove invalid FDs [12, 86, 132, 170]. An example of such method is the TANE algorithm [86], which we have already introduced in § 2.1.1. The second class include bottom-up data driven algorithms, which compare the data to find agree or difference sets and induct FDs from observation [63, 121, 166]. For example, FastFDs [166] builds upon the input data, and creates difference and agree-sets to find all minimal functional dependencies in a depth-first search. Instead of successively checking FD candidates, it searches for sets of attributes that agree on the values in certain tuple pairs. Intuitively, attribute sets that agree on certain tuple values can functionally determine only those attributes whose same tuples agree. After the agree sets are computed, it derives all valid FDs by complementing the agree sets into difference-sets and then maximizing the difference-sets to infer the FDs. The third class FD discovery algorithm combines both data-driven and schema-driven ideas to narrow down the search space. An example algorithm is HyFD [138], which adopts an alternating, two-phased discovery strategy. In a first phase, HyFD extracts a small subset of records from the input data and calculates the FDs of this non-random sample. Because only a subset of records is used in this phase, it performs particularly column-efficient. The result is a set of FDs that are either valid or almost valid with respect to the complete input dataset. In a second phase, HyFD validates the discovered FDs on the entire dataset and refines such FDs that do not yet hold. This phase is row-efficient, because it uses the previously discovered FDs to effectively prune the search space. If the validation becomes inefficient, HyFD is able to switch back into the first phase and continue there with all results discovered so far. Finally, and most recently, structure learning [105] has been applied to discovery FDs. The representative method is FDX [176]. Given a dataset, FDX proceeds in two steps: 1)it estimates the undirected form of the graph that corresponds to the FD model of the input dataset. This is done by estimating the inverse covariance matrix of the joint distribution of the random variables that correspond to our FD model; and 2) FDX finds a factorization of the inverse covariance matrix that imposes a sparse linear structure on the FD model, and thus, allows to obtain parsimonious FDs.

We note that all above methods generally assume a single dataset without considering distributed scenarios. A recent work, Distributed FastFDs [153] is to the best of our knowledge the first effort considering FD discovery over partitions, which adopts a bottom-up data driven approach. Distributed FastFDs focuses on minimizing communication cost by computing a full self-join of the dataset and computing evidence from all tuple pairs, and is expensive to implement in the secure multi-party scenario.

In our approach, we propose a secure FD validation protocol (§ 5). Secure FD validation can be boiled down to equality testing over sets and can be viewed as private set intersections over FD attributes. There exists a class of research on solving private set intersection (PSI, e.g., [101, 104]) and cardinalities (PSI-CA, e.g., [46, 160]). For example, recent work [46] explores a few PSI-CA variations and constructs several protocols. However, all existing solutions are limited to preserving the data privacy during *one-time* operation. In a full FD discovery protocol, which prunes the entire search space (§ 2.1.1), the set operations are frequently and repeatedly used, and naïvly applying them breaks their security guarantees. What's worse, to validate one FD requires set intersections for all subset of parties, and using any private set intersection techniques would require $\mathcal{O}(2^m)$ intersections given m parties. However, our approach computes the set intersection cardinality for all subsets in $O(m)$, which is optimal, since a sublinear solution cannot exist. In addition, using private set intersection would leak more information, since each party will learn the values in intersection. Our approach only outputs the set intersection cardinality for all subsets.

There is a rich literature on secure multi-party computation (e.g., see surveys [15, 119]). In the past decades, there exists extensive research on studying general purpose protocols such as garbled circuit [169], homomorphic encryption [135] and oblivious polynomial evaluation [131], and from that, building complex protocols for privacy-preserving data mining [118]. It is not clear how to use these protocols to efficiently compute PSI-CA in $\mathcal{O}(m)$. Furthermore, these protocols suffer from low scalability and inefficiency since a single FD validation would invoke multiple rounds of general protocols.

In multi-party functional dependency discovery, the privacy goal is input privacy and we would like to protect party's data from all other parties and workers. In our approach, we use mix network (mixnet, in short) [36] to setup untraceable communications. The essence of a mixnet is to provide anonymity for a batch of inputs, by changing their appearance and removing the order of arrival information. Based on the cryptographic transformations that workers do, a recent survey [152] classifies mixnets into different types such as decryption mixnet [90], and re-encryption mixnet [141] to fit different applications. Our proposed mixnets extend the literature by hybriding the decryption and re-encryption operations together per mix on each worker using the ElGamal encryption scheme [66]. Note that our mixnets use shuffling—a technique that requires at least a circuit of size $n \log n$—in $\mathcal{O}(n)$ public key operations and compare n elements from a large domain to each other, but as plaintext operations on deterministically encrypted ciphertexts. Using generic secure computation protocols, these operations would have to be performed on secret shared data.

Recent work on private database federations such as SMCQL [23] provides a practical way to answer SQL queries, which can be formed to validate FDs by selecting the count of violations and checking if the count is zero or not. SMCQL evaluates queries obliviously using Oblivious RAM [71] for secure query evaluation, and exhaustively pads dummy values for each query operator. However, with cascades of such operators, padding dummy values accumulates to a blow-up in the output size of each operator and a proportional loss in query performance [24]. SMCQL is a generic system, while our work is more specialized for operations in FD discoveries. As we will show empirically in Chapter 5, our method outperforms over method using private database federations by $1000\times$ in both computation and communication cost.

Chapter 3

APEx: Accuracy-Aware Differentially Private Data Exploration

3.1 Queries and Accuracy

In this section, we describe our query language for expressing aggregate queries and the associated accuracy measures. In the rest of the chapter, we assume that the schema and the full domain of attributes are public.

3.1.1 Exploration Queries

We consider a rich class of aggregate queries that can be expressed in a SQL-like declarative format:

BIN D ON $f(\cdot)$ WHERE $W = \{\phi_1, \ldots, \phi_L\}$
[HAVING $f(\cdot) > c$]
[ORDER BY $f(\cdot)$ LIMIT k];

Each query in our language is associated with a workload of predicates $W = \{\phi_1, \ldots, \phi_L\}$. Based on W, the tuples in a table D are divided into bins. Each bin b_i contains all the tuples in D that satisfy the corresponding predicate $\phi_i : dom(R) \rightarrow \{0, 1\}$, i.e., $b_i = \{r \in D | \phi(r) = 1\}$. As we will see later, bins need not be disjoint. Moreover, a query has an aggregation function $f : dom(R)^* \rightarrow \mathbb{R}$, which returns a numeric answer $f(b_i)$ for each bin b_i. The output of this query without the optional clauses (in square brackets) is a list of counts $f(b_i)$ for bin b_i.

Each query can be specialized using one of two optional clauses: the HAVING clause returns a list of bin identifiers b_i for which $f(b_i) > c$; and the ORDER BY ... LIMIT clause returns the k bins that have the largest values for $f(b_i)$.

In the following sections, we focus on COUNT as the aggregate function and discuss other aggregates like AVG, SUM, QUANTILE at the end (§ 3.7).

Workload Counting Query (WCQ).

BIN D ON $f(\cdot)$ WHERE $W = \{\phi_1, \ldots, \phi_L\}$;

Workload counting queries capture the large class of linear counting queries, which are the bread and butter of statistical analysis and have been the focus of majority of the work in differential privacy [81, 114]. Standard SELECT...GROUP BY queries in SQL are expressible using WCQ. For instance, consider a table D having an attribute *State* with domain {AL, AK, ..., WI, WY} and an attribute *Age* with domain $[0, \infty)$. Then, a query that returns the number of people with age above 50 for each state can be expressed using WCQ as:

BIN D ON COUNT($*$)
WHERE $W = \{Age > 50 \wedge State = \text{AL}, \ldots, Age > 50 \wedge State = \text{WY}\}$;

Other common queries captured by WCQ include: 1) *histogram queries*: the workload W partitions D into $|W_h|$ disjoint bins, e.g. $W_h = \{0 < Age \leq 10, 10 < Age \leq 20, \ldots, 90 < Age\}$ the resulting WCQ returns counts for each bin; and 2) *cumulative histograms*: we can define a workload W_p that places tuples in D into a set of inclusive bins $b_1 \subseteq b_2 \cdots \subseteq b_L$, e.g. $W_p = \{Age \leq 10, Age \leq 20, \ldots, Age \leq 90\}$. The resulting query outputs a set of cumulative counts. We call such a W_p a prefix workload.

Iceberg Counting Query (ICQ).

BIN D ON COUNT($*$) WHERE $W = \{\phi_1, \ldots, \phi_L\}$
HAVING COUNT($*$) $> c$;

An iceberg query returns bin identifiers if the aggregate value for that bin is greater than a given threshold c. For instance, a query which returns the states in the US with a population of at least 5 million can be expressed as:

BIN D ON COUNT($*$) WHERE $W = \{State = AL, ..., State = WY\}$
HAVING COUNT($*$)$>$500000000;

Note that since the answer to the query is a subset of the predicates in W (i.e., a subset of bin identifiers) but not the aggregate values for these bins, an ICQ is not a linear counting query.

Top-k Counting Query (TCQ).

BIN D ON COUNT($*$) WHERE $W = \{\phi_1, \ldots, \phi_L\}$
ORDER BY COUNT($*$) LIMIT k;

TCQ first sorts the bins based on their aggregate values (in descending order) and returns the top k bins identifiers (and not the aggregate values). For example, a query which returns the three US states with highest population can be expressed as:

BIN D ON COUNT($*$) WHERE $W = \{State = AL, ..., State = WY\}$
ORDER BY COUNT($*$) LIMIT 3;

3.1.2 Accuracy Measure

To ensure differential privacy, the answers to the exploration queries are typically noisy. To allow the data scientist to explore data with bounded error, we extend our queries to incorporate an accuracy requirement. The syntax for accuracy is inspired by that in BlinkDB [14]:

BIN D ON $f(\cdot)$ WHERE $W = \{\phi_1, \ldots, \phi_L\}$
[HAVING $f(\cdot) > c$]
[ORDER BY $f(\cdot)$ LIMIT k]
ERROR α CONFIDENCE $1 - \beta$;

We next define the semantics of the accuracy requirement for each of our query types. The accuracy requirement for a WCQ q_W is defined as a bound on the maximum error across queries in the workload W.

Definition 5 ((α, β)-WCQ accuracy)**.** *Given a workload counting query $q_W : \mathcal{D} \to \mathbb{R}^L$, where $W = \{\phi_1, \ldots, \phi_L\}$. Let $M : \mathcal{D} \to \mathbb{R}^L$ be a mechanism that outputs a vector of answers y on D. Then, M satisfies (α, β)-W accuracy, if $\forall D \in \mathcal{D}$,*

$$\Pr[\|y - q_W(D)\|_\infty \geq \alpha] \leq \beta, \tag{3.1}$$

where $\|y - q_W(D)\|_\infty = \max_j |y[i] - c_{\phi_i}(D)|$.

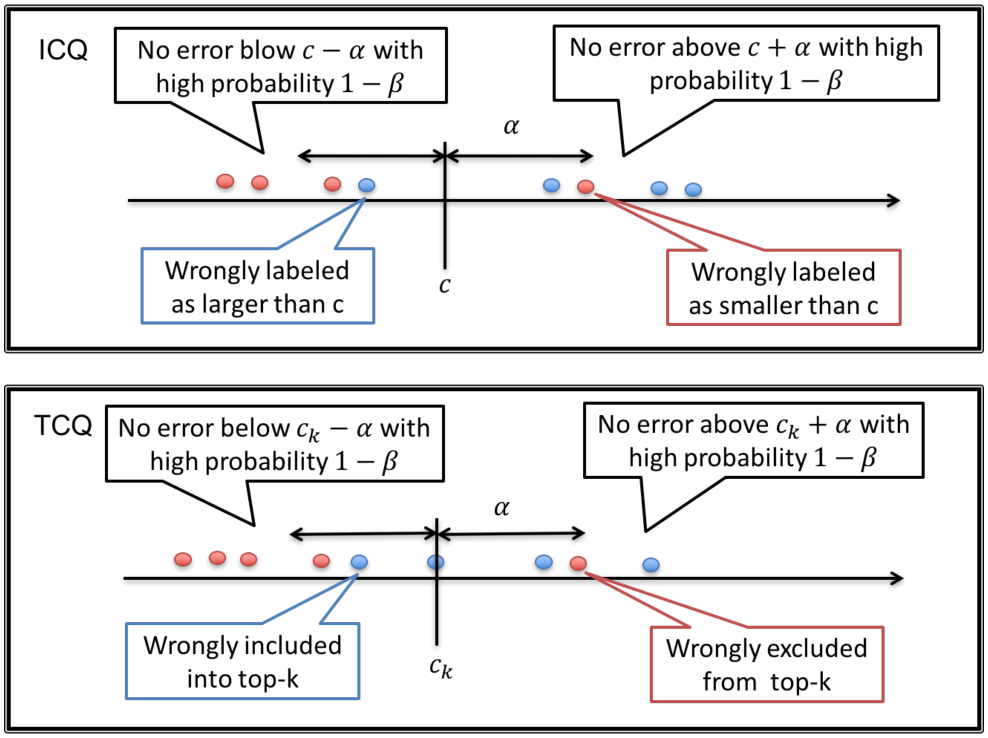

ICQ
No error blow $c - \alpha$ with high probability $1-\beta$
No error above $c + \alpha$ with high probability $1-\beta$
α
Wrongly labeled as larger than c
c
Wrongly labeled as smaller than c
TCQ
No error below $c_k - \alpha$ with high probability $1-\beta$
No error above $c_k + \alpha$ with high probability $1-\beta$
α
Wrongly included into top-k
c_k
Wrongly excluded from top-k

Figure 3.1: Accuracy requirement for ICQ and TCQ.

The output of iceberg counting queries ICQ and top-k counting queries TCQ are not numeric, but a subset of the given workload predicates. Their accuracy measures are different from WCQ, and depend on their corresponding workload counting query q_W.

Definition 6 ((α,β)-ICQ accuracy). *Given an iceberg counting query $q_{W,>c} : \mathcal{D} \to O$, where $W = \{\phi_1, \ldots, \phi_L\}$, and O is a power set of W. Let $M : \mathcal{D} \to O$ be a mechanism that outputs a subset of W. Then, M satisfies (α,β)-ICQ accuracy for $q_{W,>c}$ if for D,*

$$\Pr[|\{\phi \in M(D) \mid c_\phi(D) < c - \alpha\}| > 0] \leq \beta \tag{3.2}$$

$$\Pr[|\{\phi \in (W - M(D)) \mid c_\phi(D) > c + \alpha\}| > 0] \leq \beta \tag{3.3}$$

A mechanism for ICQ can make two kinds of errors: label predicates with true counts greater than c as $< c$ (red dots in Figure 3.1), and label predicates with true counts less than c as $> c$ (blue dots in Figure 3.1). We say a mechanism satisfies (α,β)-ICQ accuracy if with high probability, all the predicates with true counts greater than $c+\alpha$ are correctly labeled as $> c$, and all the predicates with true counts less than $c-\alpha$ are correctly labeled as $< c$. The mechanism may make arbitrary mistakes within the range $[c-\alpha, c+\alpha]$.

Definition 7 ((α, β)-TCQ accuracy)**.** *Given a top-k counting query $q_{W,k} : \mathcal{D} \to O$, where $W = \{\phi_1, \ldots, \phi_L\}$, and O is a power set of W. Let $M : \mathcal{D} \to O$ be a mechanism that outputs a subset of W. Then, M satisfies (α, β)-TCQ accuracy if for $D \in \mathcal{D}$,*

$$\Pr[|\{\phi \in M(D) \mid c_\phi(D) < c_k - \alpha\}| > 0] \leq \beta \tag{3.4}$$

$$\Pr[|\{\phi \in (\Phi - M(D)) \mid c_\phi(D) > c_k + \alpha\}| > 0] \leq \beta \tag{3.5}$$

where c_k is the k^{th} largest counting value among all the bins, and Φ is the true top-k bins.

The intuition behind Definition 7 is similar to that of ICQ and is explained in Figure 3.1: predicates with count greater than $c_k + \alpha$ are included and predicates with count less than $c_k - \alpha$ do not enter the top-k with high probability.

In the rest of the chapter, we will describe how APEx designs differentially private mechanisms to answer queries with the above defined bounds on accuracy. The advantages of our accuracy definitions are that they are intuitive (when α increases, noisier answers are expected) and we can design privacy-preserving mechanisms that introduce noise while satisfying these accuracy guarantees. On the other hand, this measure is not equivalent to other bounds on accuracy like relative error and precision/recall which can be very sensitive to small amounts of noise (when the counts are small, or when lie within a small range). For instance, if the counts of all the predicates in ICQ lie outside $[c - \alpha, c + \alpha]$, a mechanism M that perturbs counts within $\pm\alpha$ and then answers an ICQ will have precision and recall of 1.0 with high probability as it makes no mistakes. However, if all the query answers lie within $[c - \alpha, c + \alpha]$, then the precision and recall of the output of M could be 0. Incorporating other error measures like precision/recall and relative error into APEx is an interesting avenue for future work.

3.2 APEx Overview

This section outlines how APEx translates exploration queries with accuracy bounds into differentially private mechanisms, and how it ensures the privacy budget B specified by the data owner is not violated.

Accuracy Translator. Given a query (q, α, β), APEx first uses the accuracy translator to choose a mechanism M that can 1) answer q under the specified accuracy bounds, with 2) minimal privacy loss. To achieve these, APEx supports a set of ϵ-DP mechanisms that can be used to answer each query type (WCQ, ICQ, TCQ). Multiple mechanisms are supported for each query type as different mechanisms result in the least privacy loss

Algorithm 2 *APEx* Overview

```
Input: Dataset $D$, privacy budget $B$
 1: Initialize privacy loss $B_0 \leftarrow 0$, index $i \leftarrow 1$
 2: repeat
 3:     Receive $(q_i, \alpha_i, \beta_i)$ from data scientist
 4:     $\mathcal{M} \leftarrow$ mechanisms applicable to $q_i$'s type
 5:     $\mathcal{M}^* \leftarrow \{M \in \mathcal{M} \mid M.\text{TRANSLATE}(q_i, \alpha_i, \beta_i).\epsilon^u \leq B - B_{i-1}\}$
 6:     if $\mathcal{M}^* \neq \emptyset$ then
 7:         // Pessimistic Mode
 8:           $M_i \leftarrow \texttt{argmin}_{M \in \mathcal{M}^*} \; M.\text{TRANSLATE}(q_i, \alpha_i, \beta_i).\epsilon^u$
 9:         // Optimistic Mode
10:           $M_i \leftarrow \texttt{argmin}_{M \in \mathcal{M}^*} \; M.\text{TRANSLATE}(q_i, \alpha_i, \beta_i).\epsilon^l$
11:         $(\omega_i, \epsilon_i) \leftarrow M_i.\text{RUN}(q_i, \alpha_i, \beta_i, D)$
12:         $B_i \leftarrow B_{i-1} + \epsilon_i$, $i$++
13:         return $\omega_i$
14:     else
15:         $B_i = B_{i-1}$, $i$++
16:         return 'Query Denied'
17:     end if
18: until No more queries sent by local exploration
```

depending on the query and the dataset (as shown theoretically and empirically in § 3.3 and § 3.5, respectively).

Each mechanism M exposes two functions: M.TRANSLATE, which *translates* a query and accuracy requirement into a lower and upper bound (ϵ^l, ϵ^u) on the privacy loss if M is executed, and M.RUN that runs the differentially private algorithm and returns an approximate answer ω for the query. The answer ω is guaranteed to satisfy the specified accuracy requirement. Moreover, M satisfies ϵ^u differential privacy. The mechanisms supported by *APEx* and the corresponding TRANSLATE functions are described in § 3.3. In some cases (e.g., Algorithm 5 in § 3.3.3), the privacy loss incurred by M may be $\epsilon \in (\epsilon^l, \epsilon^u)$ that is smaller than the worst case, depending on the characteristics of the dataset.

As described in Algorithm 2, *APEx* first identifies the mechanisms $\mathcal{M}$ that are applicable for the type of the query q_i (Line 4). Next, it runs M.TRANSLATE to get conservative estimates on privacy loss ϵ^u for all these mechanisms (Line 5). *APEx* picks one of the mechanisms M from those that can be safely run using the remaining privacy budget, executes M.RUN, and returns the output to the data scientist. As we will see there exist mechanisms where the privacy loss can vary based on the data in a range between $[\epsilon^l, \epsilon^u]$, and the actual privacy loss is unknown before running the mechanism. In such cases, $\mathcal{A}$PEx can choose to be *pessimistic* and pick the mechanism with the least ϵ^u (Line 8), or choose to be *optimistic* and pick the mechanism with the least ϵ^l (Line 10).

Privacy Analyzer. Given a sequence of queries $(M_1, \ldots, M_i)$ already executed by the privacy engine that satisfy an overall B_{i-1}-differential privacy and a new query (q_i, α_i, β_i), $\mathcal{A}$PEx identifies a set of mechanisms $\mathcal{M}^*$ that all will have a worst case privacy loss smaller than $B - B_{i-1}$ (Line 5). That is, running any mechanism in $\mathcal{M}^*$ will not result in exceeding the privacy budget in the worst case. If $\mathcal{M}^* = \emptyset$, then $\mathcal{A}$PEx returns 'Query Denied' to the data scientist (Line 16). Otherwise, $\mathcal{A}$PEx runs one of the mechanisms M_i from $\mathcal{M}^*$ by executing $M_i.\textsc{run}()$ and the output ω_i will be returned to the data scientist. $\mathcal{A}$PEx then increments B_{i-1} by the actual privacy loss ϵ_i rather than the upperbound ϵ^u (Line 12). As explained above, in some cases $\epsilon_i < \epsilon^u$ as different execution paths in the mechanism can have different privacy loss. Nevertheless, the privacy analyzer guarantees that the execution of any sequence of mechanisms $(M_1, M_2, \ldots, M_i)$ before it halts is B-differentially private (see § 3.4, Theorem 8).

3.3 Accuracy Translator

In this section, we present the accuracy-to-privacy translation mechanisms supported by $\mathcal{A}$PEx and the corresponding RUN and TRANSLATE functions. We first present a general baseline mechanism for all three types of exploration queries including workload counting query (WCQ), iceberg counting query (ICQ), and top-k counting query (TCQ). Then we show specialized mechanisms for each type of exploration queries, which consumes smaller privacy cost under different scenarios than the baseline.

We represent the workload in these three queries in a matrix form, like the prior work for WCQ [113, 114, 123]. There are many possible ways to transform a workload into a matrix. Given a workload counting query q_W with the set of predicates $W = \{\phi_1, \ldots, \phi_L\}$, the full domain of the relation $dom(R)$ is partitioned based on W to form the new discretized domain $dom_W(R)$ such that any predicate $\phi_i \in W$ can be expressed as a union of partitions in the new domain $dom_W(R)$ and the number of partitions is minimized. For example, given $W = \{Age > 50 \wedge State = \text{AL}, \ldots, Age > 50 \wedge State = \text{WY}\}$, one possible partition is $dom_W(R) = \{Age > 50 \wedge State = \text{AL}, \ldots, Age > 50 \wedge State = \text{WY}, Age \leq 50\}$.

Let $\mathbf{x}$ represent the histogram of the table D over $dom_W(R)$. The set of corresponding counting queries $\{c_{\phi_1}, \ldots, c_{\phi_L}\}$ for q_W can be represented by a matrix $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_L]^T$ of size $L \times |dom_W(R)|$. Hence, the answer to each counting query is $c_{\phi_i}(D) = \mathbf{w}_i \cdot \mathbf{x}$

and the answer to the workload counting query is simply $\mathbf{Wx}$. We denote and use this transformation by $\mathbf{W} \leftarrow \mathcal{T}(W), \mathbf{x} \leftarrow \mathcal{T}_W(D)$. Unlike prior work [113, 114, 123] which aims to bound the expected total error for one query, APEx aims to bound the maximum error per query with high probability which is more intuitive in the process of data exploration.

3.3.1 Baseline Translation

The baseline translation for all three query types is based on the Laplace mechanism [53, 56], a classic and widely used differentially private algorithm, which can be used for WCQ, ICQ, and TCQ. Formally,

Definition 8 (Laplace Mechanism (Vector Form) [53, 56])**.** *Given an $L \times |dom_W(R)|$ query matrix $\mathbf{W}$, the randomized algorithm LM that outputs the following vector is ϵ-differentially private: $LM(\mathbf{W}, \mathbf{x}) = \mathbf{Wx} + Lap(b_{\mathbf{W}})^L$ where $b_{\mathbf{W}} = \frac{\|\mathbf{W}\|_1}{\epsilon}$, and $Lap(b)^L$ denote a vector of L independent samples η_i from a Laplace distribution with mean 0 and variance $2b^2$, i.e., $\Pr[\eta_i = z] \propto e^{-z/b}$ for $i = 1, \ldots, L$.*

The constant $\|\mathbf{W}\|_1$ is equal to the sensitivity of queries set defined by the workload $\mathbf{W}$ [113, 114]. It measures the maximum difference in the answers to the queries in $\mathbf{W}$ on any two databases that differ only a single record. Mathematically, it is the maximum of L_1 norm of a column of $\mathbf{W}$.

Algorithm 3 provides the RUN and TRANSLATE of Laplace mechanism for all three query types. This algorithm first transforms the query q_W and the data D into matrix representation $\mathbf{W}$ and $\mathbf{x}$. The TRANSLATE outputs a lower and upper bound (ϵ^l, ϵ^u) for each query type with a given accuracy requirement and these two bounds are the same as Laplace mechanism is data independent. However, these bounds vary among query types. The RUN takes the privacy budget computed by TRANSLATE(q, α, β) (Line 3), and adds the corresponding Laplace noise $[\tilde{x}_1, \ldots, \tilde{x}_L]$ to the true workload counts $\mathbf{Wx}$. When q is a WCQ, the noisy counts are returned directly; when q is an ICQ, the bin ids (the predicates) that have noisy counts $\geq c$ are returned; when q is a TCQ, the bin ids (the predicates) that have the largest k noisy counts are returned. Beside the noisy output, the privacy budget consumed by this mechanism is returned as well. The following theorem summarizes the properties of the two functions RUN and TRANSLATE.

Theorem 3. *Given a query q where $q.type \in \{WCQ, ICQ, TCQ\}$, Laplace mechanism (Algorithm 3) denoted by M can achieve (α, β)-$q.type$ accuracy by executing the function* RUN(q, α, β, D) *for any $D \in \mathcal{D}$, and satisfy differential privacy with a minimal cost of* TRANSLATE$(q, \alpha, \beta).\epsilon^u$.

Algorithm 3 Laplace Mechanism (LM) (q, α, β, D)

```
1: Initialize $\mathbf{W} \leftarrow \mathcal{T}(W = \{\phi_1, \ldots, \phi_L\}), \mathbf{x} \leftarrow \mathcal{T}_W(D), \alpha, \beta$
2: function RUN($q, \alpha, \beta, D$)
3:     $\epsilon \leftarrow$ TRANSLATE$(q_W, \alpha, \beta).\epsilon^u$
4:     $[\tilde{x}_1, \ldots, \tilde{x}_L] \leftarrow \mathbf{W}\mathbf{x} + Lap(b)^L$, where $b = \|\mathbf{W}\|_1/\epsilon$
5:     if $q$.type==WCQ (i.e., $q_W$) then
6:         return $([\tilde{x}_1, \ldots, \tilde{x}_L], \epsilon)$
7:     else if $q$.type==ICQ (i.e., $q_{W,>c}$) then
8:         return $(\{\phi_i \in W \mid \tilde{x}_i > c\}, \epsilon)$
9:     else if $q$.type==TCQ (i.e., $q_{W,k}$) then
10:        return $(\texttt{argmax}^k_{\phi_1,\ldots,\phi_L} \tilde{x}_i, \epsilon)$
11:    end if
12: end function
13: function TRANSLATE($q, \alpha, \beta$)
14:    if $q$.type==WCQ (i.e., $q_W$) then
15:        return $(\epsilon^u = \frac{\|\mathbf{W}\|_1 \ln(1/(1-(1-\beta)^{1/L}))}{\alpha}, \epsilon^l = \epsilon^u)$
16:    else if $q$.type==ICQ (i.e., $q_{W,>c}$) then
17:        return $(\epsilon^u = \frac{\|\mathbf{W}\|_1(\ln(1/(1-(1-\beta)^{1/L}))-\ln 2)}{\alpha}, \epsilon^l = \epsilon^u)$
18:    else if $q$.type==TCQ (i.e., $q_{W,k}$) then
19:        return $(\epsilon^u = \frac{\|\mathbf{W}\|_1 2(\ln(L/(2\beta)))}{\alpha}, \epsilon^l = \epsilon^u)$
20:    end if
21: end function
```

Proof. We first prove privacy for each of the three types of queries, and then prove the accuracy guarantee.

Privacy Proof for WCQ

PROOF: For any pair of neighbouring databases D and D' such that $|D \backslash D' \cup D' \backslash D| = 1$, given a WCQ query q_W, LM adds noise drawn from $Lap(b)$ to query c_{ϕ_i}, where $b = \|\mathbf{W}\|_1/\epsilon$.

Consider a column vector of counts $y = [y_1, \cdots, y_L]$,

$$\begin{aligned}
&\frac{\Pr[q_{\mathbf{W}}(D) + Lap(b_{\mathbf{W}})^L = y]}{\Pr[q_{\mathbf{W}}(D') + Lap(b_{\mathbf{W}})^L = y]} \\
= &\quad \frac{\prod_1^L \Pr[c_{\phi_i}(D) + \eta_i = y_i]}{\prod_1^L \Pr[c_{\phi_i}(D') + \eta_i = y_i]} = \prod_1^L \frac{exp(\frac{-\epsilon|y_i - c_{\phi_i}(D)|}{\|\mathbf{W}\|_1})}{exp(\frac{-\epsilon|y_i - c_{\phi_i}(D')|}{\|\mathbf{W}\|_1})} \\
= &\quad exp(\frac{\epsilon}{\|\mathbf{W}\|_1} \sum_1^L (|y_i - c_{\phi_i}(D')| - |y_i - c_{\phi_i}(D)|)) \\
\leq &\quad exp(\frac{\epsilon}{\|\mathbf{W}\|_1} \sum_1^L (|c_{\phi_i}(D) - c_{\phi_i}(D')|)) = exp(\epsilon)
\end{aligned}$$

Therefore, it satisfies $\epsilon = \|\mathbf{W}\|_1/b$-differential privacy. □

Privacy Proof for ICQ and TCQ

PROOF: For ICQ (and TCQ, respectively), Line 8 (Line 10) post-processes the noisy answers only without accessing the true data. By the post-processing property of differential privacy [52], Laplace mechanism for ICQ (TCQ) satisfies $\epsilon = \|\mathbf{W}\|_1/b$-differential privacy. □

Accuracy Proof for WCQ

PROOF: Give a WCQ q_W, for any $D \in \mathcal{D}$, setting $b \leq \frac{\alpha}{\ln(1/(1-(1-\beta)^{1/L}))}$ bounds the failing probability, i.e.,

$$\begin{aligned}
&\Pr[\|LM(\mathbf{W}, \mathbf{x}) - q_W(D)\|_\infty \geq \alpha] \\
= &\quad 1 - \prod_{i \in [1,L]} (1 - \Pr[|\eta_i| > \alpha]) = 1 - (1 - e^{-\alpha/b})^L \leq \beta \quad \square
\end{aligned}$$

Accuracy Proof for ICQ

PROOF: Given ICQ $q_{W,>c}$, for any $D \in \mathcal{D}$, setting $b \leq \frac{\alpha}{\ln(1/(1-(1-\beta)^{1/L}))-\ln 2}$ bounds the failure probability, i.e.,

$$\begin{aligned}
&\Pr[|\{\phi \in M(D) \mid c_\phi(D) < c - \alpha\}| > 0] \\
= &\quad 1 - \prod_{\phi \in W : c_\phi(D) - c + \alpha < 0} (1 - \Pr[c_\phi(D) - c + \eta > 0]) \\
\leq &\quad 1 - \prod_{\phi \in W : c_\phi(D) - c + \alpha < 0} (1 - \Pr[\eta > \alpha]) \\
\leq &\quad 1 - (1 - e^{-\alpha/b}/2)^L < \beta
\end{aligned}$$

The proof for the other condition is analogous. $\square$

Accuracy Proof for TCQ

PROOF: Given a TCQ $q_{W,k}$, for any $D \in \mathcal{D}$, W.L.O.G. let $c_{\phi_1}(D) \geq \cdots \geq c_{\phi_k}(D) \cdots \geq c_L(D)$ and the noise added to these counts be $\eta_1, \ldots, \eta_L$ respectively. Let $c_k = c_{\phi_k}(D)$ the k^{th} largest counting value. Setting noise parameter $b \leq \frac{\alpha}{2\ln(L/(2\beta))}$ bounds the failing probability, i.e.,

$$\begin{aligned}
& \Pr[|\{\phi \in M(D) \mid c_\phi(D) < c_k - \alpha\}| > 0] \\
\leq\ & 1 - \Pr[(\max_{i>k, c_{\phi_i}(D) < c_k - \alpha} \eta_i < \frac{\alpha}{2}) \wedge (\min_{i \leq k} \eta_i > -\frac{\alpha}{2})] \\
=\ & \Pr[(\max_{i>k, c_{\phi_i}(D) < c_k - \alpha} \eta_i \geq \frac{\alpha}{2}) \vee (\min_{i \leq k} \eta_i \geq -\frac{\alpha}{2})] \\
\leq\ & (L-k)e^{-\alpha/(2b)}/2 + ke^{-\alpha/(2b)}/2 = Le^{\alpha/(2b)}/2 \leq \beta
\end{aligned} \tag{3.6}$$

The proof for $\Pr[|\{\phi \in (\Phi - M(D)) \mid c_\phi(D) > c_k + \alpha\}| > 0] \leq \beta$ is analogous. $\square$

3.3.2 Special Translation for WCQ

The privacy cost of the Laplace mechanism increases linearly with $\|\mathbf{W}\|_1$, which is the sensitivity of the workload in the query. For example, a prefix workload has a sensitivity equals to the workload size L. When L is very large, the privacy cost grows drastically. To address this problem, APEx provides a special translation for WCQ, called *strategy-based mechanism*. This mechanism considers a different *strategy* workload $\mathbf{A}$ such that 1) $\mathbf{A}$ has a low sensitivity $\|\mathbf{A}\|_1$; and 2) rows in $\mathbf{W}$ can be reconstructed using a small number of rows in $\mathbf{A}$.

Let strategy matrix $\mathbf{A}$ be a $l \times |dom_W(R)|$ matrix, and $\mathbf{A}^+$ denote its Moore-Penrose pseudoinverse, such that $\mathbf{W}\mathbf{A}\mathbf{A}^+ = \mathbf{W}$. Given such a strategy workload $\mathbf{A}$, we can first answer $\mathbf{A}$ using Laplace mechanism (i.e., $\hat{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\eta}$, where $\boldsymbol{\eta} \sim Lap(b)^l$ and $b = \frac{\|\mathbf{A}\|_1}{\epsilon}$), and then reconstruct answers to $\mathbf{W}$ from the noisy answers to $\mathbf{A}$ as a postprocessing step (i.e., $(\mathbf{W}\mathbf{A}^+)\hat{y}$). This approach is formally known as the Matrix mechanism [113, 114] and shown in the RUN of Algorithm 4. If a strategy $\mathbf{A}$ is used for this mechanism, we denote it by $\mathbf{A}$-strategy mechanism. In this thesis, we consider several popular strategies in prior work [113, 114, 123], such as hierarchical matrix H_2. Techniques like HDMM [123] can automatically solve for a good strategy (but this is not our focus).

However, translating the accuracy requirement on WCQ-SM is nontrivial as the answers to the query $q_W(D)$ are linear combinations of noisy answers. The errors are due to the

Algorithm 4 WCQ-SM $(q_W, \alpha, \beta, D, \mathbf{A})$

```
1: Initialize $\mathbf{W} \leftarrow \mathcal{T}(W), \mathbf{x} \leftarrow \mathcal{T}_W(D), \alpha, \beta, \mathbf{A}$
2: function RUN($q_W, \alpha, \beta, D$)
3:     $\epsilon \leftarrow$ TRANSLATE($\mathbf{W}, \alpha, \beta$).$\epsilon^u$
4:     $\omega \leftarrow \mathbf{WA}^+(\mathbf{Ax} + Lap(b)^l)$, where $b = \|A\|_1/\epsilon$
5:     return $(\omega, \epsilon)$
6: end function
7: function TRANSLATE($q_W, \alpha, \beta$)
8:     Set $u = \frac{\|\mathbf{A}\|_1 \|\mathbf{WA}^+\|_F}{\alpha\sqrt{\beta/2}}$ and $l = 0$
9:     $\epsilon$ = BINARYSEARCH($l, u$, ESTIMATEBETA($\cdot, \alpha, \beta, \mathbf{WA}^+$))
10:    return $(\epsilon^u = \epsilon, \epsilon^l = \epsilon)$
11: end function
12: function ESTIMATEBETA($\epsilon, \alpha, \beta, \mathbf{WA}^+$)
13:     Sample size $N = 10000$ and failure counter $n_f = 0$
14:     for $i \in [1, \ldots, N]$ do
15:         Sample noise $\boldsymbol{\eta}_i \sim Lap(\|\mathbf{A}\|_1/\epsilon)^l$
16:         if $\|(\mathbf{WA}^+)\boldsymbol{\eta}_i\|_\infty > \alpha$ then
17:             $n_f$++
18:         end if
19:     end for
20:     $\beta_e = n_f/N$, $p = \beta/100$
21:     $\delta\beta = z_{1-p/2}\sqrt{\beta_e(1-\beta_e)/N}$
22:     return $(\beta_e + \delta\beta + p/2) < \beta$
23: end function
```

sums of weighted Laplace random variables which have non-trivial CDFs for the accuracy translation. Hence, we propose an accuracy to privacy translation method shown in the TRANSLATE function of Algorithm 4. We first set an upper bound u for the privacy to achieve (α, β)-WCQ accuracy based on Theorem 4 and conduct a binary search on a privacy cost ϵ between l and u (Line 9) such that the failing probability to bound the error by α equals to β. During this binary search, for each ϵ between l and u, we run Monte Carlo simulation to learn the empirical failing rate β_e to bound the error by α shown in the function ESTIMATEBETA() such that with high confidence $1-p$, the true failing probability β_t lies within $\beta_e \pm \delta\beta$. If the empirical failing rate β_e is sufficiently smaller than β, then the upper bound is set to ϵ; otherwise the lower bound is set to ϵ. The next value for ϵ is $(l+u)/2$. This search process stops when l and u is sufficiently small. This approach can be generalized to all data-independent differentially private mechanisms. The simulation can be done offline. We state the results in Theorem 5.

Theorem 4. *Given a WCQ $q_W : \mathcal{D} \to \mathbb{R}^L$, for a table $D \in \mathcal{D}$, let $(\mathbf{W}, \mathbf{x}) = \mathcal{T}(W, D)$. Let $\mathbf{A}$ be a strategy used in Algorithm 4 to answer q_W. When $\epsilon \geq \frac{\|\mathbf{A}\|_1 \|\mathbf{W}\mathbf{A}^+\|_F}{\alpha\sqrt{\beta/2}}$, $\mathbf{A}$-strategy mechanism achieves (α, β)-WCQ accuracy.*

Proof. The noise vector added to the final query answer of $q_W(\cdot)$ using $\mathbf{A}$-strategy mechanism is $\hat{\boldsymbol{\eta}} = [\hat{\eta}_1, \ldots, \hat{\eta}_L] = (\mathbf{W}\mathbf{A}^+)\boldsymbol{\eta}$. Each noise variable $\hat{\eta}_i$ has a mean of 0, and a variance of $\sigma_i^2 = c_i \cdot (2b^2)$, where $c_i = \sum_{j=1}^{l} (\mathbf{W}\mathbf{A}^+)[i,j]^2$, and hence $\Pr[|\hat{\eta}_i| \geq \alpha] \leq \frac{2c_i b_{\mathbf{A}}^2}{\alpha^2}$ by Chebyshev's inequality. By union bound, the failing probability is bounded by

$$\begin{aligned} & \Pr[\|LM(\mathbf{W}, \mathbf{x}) - q_W(D)\|_\infty \geq \alpha] \\ = \quad & \Pr[\cup_{i \in [1,L]} |\eta_i| \geq \alpha] \leq \sum_{i \in [1,L]} \frac{2c_i b^2}{\alpha^2} \leq \beta \end{aligned}$$

It requires $b \leq \frac{\alpha\sqrt{\beta/2}}{\|\mathbf{W}\mathbf{A}^+\|_F}$, hence $\epsilon \geq \|A\|_1 / b \geq \frac{\|\mathbf{A}\|_1 \|\mathbf{W}\mathbf{A}^+\|_F}{\alpha\sqrt{\beta/2}}$. □

Theorem 5. *Given a workload counting query $q_W : \mathcal{D} \to \mathbb{R}^L$. Answering q_W with $\mathbf{A}$-strategy mechanism by executing* RUN(q_W, α, β, D) *in Algorithm 4 achieves (α, β)-WCQ accuracy for any $D \in \mathcal{D}$ with an approximated minimal privacy cost of* TRANSLATE$(q_W, \alpha, \beta).\epsilon^u$.

Proof. Given an ϵ, the simulation in the function ESTIMATEFAILINGRATEMC() ensures that with high probability $1 - p$, the true failing probability β_t to bound $\|(\mathbf{W}\mathbf{A}^+)\eta\|_\infty$ by α lies within $\beta_e \pm \delta\beta$. The failing probability to bound $\beta_t < \beta_e + \delta\beta$ is $p/2$. By union bound, ϵ ensures (α, β')-WCQ accuracy, where $\beta' < \beta + \delta\beta + p/2$. If $(\beta + \delta\beta)(1 - p/2) + p/2 < \beta + \delta\beta + p/2 < \beta$, then this ϵ ensures (α, β)-WCQ accuracy. Beside this estimation, in the binary search of TRANSLATE(), we stop when $\epsilon_{\min}$ and $\epsilon_{\max}$ are sufficiently close. Hence, the privacy cost returned by TRANSLATE() is an approximated minimal privacy cost required for (α, β)-WCQ accuracy. □

3.3.3 Special Translation for ICQ

The strategy-based mechanism that we used for WCQ can be adapted to answer ICQ if used in conjunction with a post-processing step. We also present another novel data dependent translation strategy for ICQ that may result in different privacy loss for different datasets given the same accuracy requirement.

Strategy-based Mechanism (ICQ-SM)

The data scientist can pose a workload counting query q_W with (α, β)-WCQ requirement via APEx, and then use the noisy answer of $q_W(D)$ to learn $q_{W,>c}(D)$ locally. This corresponds to a post-processing step of a differentially private mechanism and hence still ensures the same level of differential privacy guarantee. On the other hand, (α, β)-ICQ accuracy only requires to bound one-sided noise by α with probability $(1 - \beta)$, and (α, β)-WCQ accuracy requires to bound two-sided noise with the same probability. Hence, if a mechanism has a failing probability of β to bound the error for WCQ, then using the same mechanism has a failing probability of $\beta/2$ to bound the error for ICQ.

Multi-Poking Mechanism (ICQ-MPM)

We propose a data-dependent translation for ICQ, which can be used as a subroutine for mechanisms that involve threshold testing. For ease of explanation, we will illustrate this translation with a special case of ICQ when the workload size $L = 1$, denoted by, $q_{\phi,>c}(\cdot)$. Intuitively, when $c_\phi(D)$ is much larger (or smaller) than c, then a much larger (smaller resp.) noise can be added to $c_\phi(D)$ without changing the output of APEx.

Example 5: Consider a query $q_{\phi,>c}$, where $c = 100$. To achieve (α, β) accuracy for this query, where $\alpha = 10, \beta = 0.1^{10}$, the Laplace mechanism requires a privacy cost of $\frac{\ln(1/(2\beta))}{\alpha} = 2.23$ by Theorem 3, regardless of input D. Suppose $c_\phi(D) = 1000$. In this case, $c_\phi(D)$ is much larger than the threshold c, and the difference is $\frac{(1000-100)}{\alpha} = 90$ times of the accuracy bound $\alpha = 10$. Hence, even when applying Laplace comparison mechanism with a privacy cost equals to $\frac{2.23}{90} \approx 0.25$ wherein the noise added is bounded by 90α with high probability $1 - \beta$, the noisy difference $c_\phi(D) - c + \eta_{sign}$ will still be greater than 0 with high probability. □

This is an example where a different mechanism rather than Laplace mechanism achieves the same accuracy with a smaller privacy cost. Note that the tightening of the privacy cost in this example requires to know the value of $c_\phi(D)$. It is difficult to determine a privacy budget for poking without looking at the query answer. To tackle this challenge, we propose an alternative approach that allows of m *pokes* with increasing privacy cost. This approach is summarized in Algorithm 5 as Multi-Poking Mechanism (MPM). This approach first computes the privacy cost if all m pokes are needed, $\epsilon_{\max} = \frac{\ln(m/(2\beta))}{\alpha}$. The first poke checks if bins have either sufficiently large noisy differences $\tilde{\mathbf{y}}$ with respect to the accuracy α_0 for the current privacy cost (Lines 7-9). If this is true (Line 10), then the set of predicates with sufficiently large positive differences is returned; otherwise, the privacy

Algorithm 5 ICQ-MPM($q_{W,>c}, \alpha, \beta, D, m$)

```
1: Initialize $\mathbf{W} \leftarrow \mathcal{T}(W), \mathbf{x} \leftarrow \mathcal{T}_W(D), \alpha, \beta, m = 10$
2: function RUN($q_{W,>c}, \alpha, \beta, D$)
3:     Compute $\epsilon_{\max} = \text{TRANSLATE}(q_{W,>c}, \alpha, \beta).\epsilon^u$
4:     Initial privacy cost $\epsilon_0 = \epsilon_{\max}/m$
5:     $\tilde{\mathbf{y}}_0 = \mathbf{W}\mathbf{x} - c + \boldsymbol{\eta}_0$, where $\boldsymbol{\eta}_0 \sim Lap(\|\mathbf{W}\|_1/\epsilon_0)^L$
6:     for $i = 0, 1, \ldots, m-2$ do
7:         Set $\alpha_i = \|\mathbf{W}\|_1 \ln(mL/(2\beta))/\epsilon_i$
8:         $W_+ \leftarrow \{\phi_j \in W \mid (\tilde{\mathbf{y}}_i[j] - \alpha_i)/\alpha \geq -1\}$
9:         $W_- \leftarrow \{\phi_j \in W \mid (\tilde{\mathbf{y}}_i[j] + \alpha_i)/\alpha \leq 1\}$
10:        if $(W_+ \cup W_-) = W$ then
11:            return $(W_+, \epsilon_i)$
12:        else
13:            Increase privacy budget $\epsilon_{i+1} = \epsilon_i + \epsilon_{\max}/m$
14:            for $j = 1, \ldots, L$ do
15:                $\boldsymbol{\eta}_{i+1}[j] = \text{RelaxPrivacy}(\boldsymbol{\eta}_i[j], \epsilon_i, \epsilon_{i+1})$ [107]
16:            end for
17:            New noisy difference $\tilde{\mathbf{y}}_{i+1} = \mathbf{W}\mathbf{x} - c + \boldsymbol{\eta}_{i+1}$
18:        end if
19:    end for
20:    return $(\{\phi_j \in W \mid \tilde{\mathbf{y}}_{m-1}[j] > 0\}, \epsilon_{\max})$
21: end function
22: function TRANSLATE($q_{W,>c}, \alpha, \beta$)
23:     return $\epsilon^u = \frac{\|\mathbf{W}\|_1 \ln(mL/(2\beta))}{\alpha}, \epsilon^l = \frac{\epsilon^u}{m}$
24: end function
```

budget is relaxed with additional $\epsilon_{\max}/m$. At $(i+1)$th iteration, instead of sampling independent noise, we apply the RelaxPrivacy algorithm [108] to correlate the new noise $\boldsymbol{\eta}_{i+1}$ with noise $\boldsymbol{\eta}_i$ from the previous iteration. In this way, the privacy loss of the first $i+1$ iterations is ϵ_{i+1}, and the noise added in the $i+1$th iteration is equivalent to a noise generated with Laplace distribution with privacy parameter $b = (1/\epsilon_{i+1})$. This approach allows the data scientist to learn the query answer with a gradual relaxation of privacy cost. This process repeats until all $\epsilon_{\max}$ is spent. We show that Algorithm 5 achieves both accuracy and privacy requirements.

Theorem 6. *Given a query $q_{W,>c}$, Multi-Poking Mechanism (Algorithm 5), achieves (α, β)-ICQ accuracy by executing function* RUN$(q_{W,>c}, \alpha, \beta, D)$*, with differential privacy of* TRANSLATE$(q_{W,>c}, \alpha, \beta).\epsilon^u$.

Proof. For each $\phi \in W$, (i) when $q_\phi(D) < c - \alpha$,

$$\begin{aligned}
& \Pr[\phi \in MPM^{\alpha,\beta}_{q_\phi,>c}(D) \mid c_\phi(D) < c - \alpha] \\
= & \sum_{i=0}^{m-1} \Pr[c_\phi(D) - c + \eta - \alpha_i + \alpha > 0 \mid c_\phi(D) - c + \alpha < 0] \\
< & \sum_{i=0}^{m-1} \Pr[\eta_i > \alpha_i] = \sum_{i=0}^{m-1} e^{-\alpha_i \epsilon_i / \|\mathbf{W}\|_1}/2 = m \cdot \beta/(mL) = \beta/L
\end{aligned}$$

and (ii) when $q_\phi(D) < c + \alpha$,

$$\begin{aligned}
& \Pr[\phi \notin MPM^{\alpha,\beta}_{\phi,>c}(D) \mid c_\phi(D) > c + \alpha] \\
= & \sum_{i=0}^{m-1} \Pr[c_\phi(D) - c + \eta + \alpha_i - \alpha < 0 \mid c_\phi(D) - c - \alpha > 0] \\
< & \sum_{i=0}^{m-1} \Pr[\eta_i < -\alpha_i] = \sum_{i=0}^{m-1} e^{-\alpha_i \epsilon_i / \|\mathbf{W}\|_1}/2 = m \cdot \beta/(mL) = \beta/L
\end{aligned}$$

As $|W| = L$, the failing probability is bounded by β.

The RelaxPrivacy algorithm [108] correlates new noise η_{i+1} with noise η_i from the previous iteration. In this way, the composition of the first $i+1$ iterations is ϵ_{i+1}, and the noise added in the $i+1$th iteration is equivalent to a noise generated with Laplace distribution with privacy budget ϵ_{i+1} and the first i iterations also satisfy ϵ_i-DP for $i = 0, 1, \ldots, m-1$. □

The privacy loss of multi-poking mechanism at the worst case (the value returned by TRANSLATE) is greater than that of the baseline LM, but this mechanism may stop before $\epsilon_{\max}$ is used up, and hence it potentially saves privacy budget for the subsequent queries.

3.3.4 Special Translation for TCQ

This section provides a translation mechanism, known as Laplace top-k Mechanism (shown in Algorithm 6). This mechanism is a generalized *report-noisy-max* algorithm [56]: when $k = 1$, it adds noise drawn from $Lap(1/\epsilon)$ to all queries, and only reports the query number that has the maximum noisy count (and not the noisy count). When $k \geq 1$, this mechanism first perturbs $\mathbf{Wx}$ with Laplace noise $\eta \sim Lap(b)^L$, where $b = k/\epsilon$. These predicates are then sorted based on their corresponding noisy counts in descending order, and the first k boolean formulae are outputted. The privacy cost is summarized in Theorem 7 and the proof follows that of the report-noisy-max algorithm.

Theorem 7. *Given a top-k counting query $q_{W,k}(\cdot)$, where $W = \{\phi_1, \ldots, \phi_L\}$, for a table $D \in \mathcal{D}$, Laplace top-k mechanism (Algorithm 6) denoted by $LTM^{\alpha,\beta}_{W,k}(\cdot)$, can achieve (α, β)-TCQ accuracy by executing* RUN$(q_{W,k}, \alpha, \beta, D)$ *with minimal differential privacy cost of* TRANSLATE$(q_{W,k}, \alpha, \beta).\epsilon^u$.

Proof. Fix $D = D' \cup \{t\}$. Let $(x_1, \ldots, x_L)$, respectively $(x'_1, \ldots, x'_L)$, denote the vector of answers to the set of linear counting queries $q_{\phi_1}, \ldots, q_{\phi_L}$ when the table is D, respectively D'. Two properties are used: (1) Monotonicity of Counts: for all $j \in [L]$, $x_j \geq x'_j$; and (2) Lipschitz Property: for all $j \in [L]$, $1 + x'_j \geq x_j$.

Fix any k different values $(i_1, \ldots, i_k)$ from $[L]$, and fix noises $(\eta_{i_{k+1}}, \ldots, \eta_{i_L})$ drawn from $Lap(k/\epsilon)^{L-k}$ used for the remaining linear counting queries. Given these fixed noises, for $l \in \{i_1, \ldots, i_k\}$, we define

$$\eta^*_l = \min_{\eta} : (x_l + \eta > (\max_{j \in i_{k+1}, \ldots, i_L} x_j + \eta_j)) \tag{3.7}$$

For each $l \in \{i_1, \ldots, i_k\}$, we have

$$\begin{aligned} x'_l + (1 + \eta^*_l) = (1 + x'_l) + \eta^*_l \geq x_l + \eta^*_l \\ > \max_{j \in i_{k+1}, \ldots, i_L} x_j + \eta_j \geq \max_{j \in i_{k+1}, \ldots, i_L} x'_j + \eta_j \end{aligned}$$

Hence, if $\eta_l \geq r^*_l + 1$ for all $l \in \{i_1, \ldots, i_k\}$, then $(i_1, \ldots, i_k)$ will be the output when the table is D' and the noise vector is $(\eta_{i_1}, \ldots, \eta_{i_k}, \ldots, \eta_L)$. The probabilities below are over the choices of $(\eta_{i_1}, \ldots, \eta_{i_k}) \sim Lap(k/\epsilon)^k$.

$$\begin{aligned} & \Pr[(i_1, \ldots, i_k) \mid D', \eta_{i_{i+1}}, \ldots, \eta_{i_L}] \\ \geq & \prod_{l \in \{i_1, \ldots, i_k\}} \Pr[\eta_l \geq 1 + \eta^*_l] \ \geq \prod_{l \in \{i_1, \ldots, i_k\}} e^{-k/\epsilon} \Pr[\eta_l \geq \eta^*_l] \\ \geq & \ e^{-\epsilon} \Pr[(i_1, \ldots, i_k) \mid D, \eta_{i_{i+1}}, \ldots, \eta_{i_L}] \end{aligned} \tag{3.8}$$

Proof of the other direction follows analogously.

Therefore, ϵ-differential privacy is guaranteed. □

Algorithm 6 TCQ-LTM($q_{W,k}, \alpha, \beta, D$))

```
1: Initialize $\mathbf{W} \leftarrow \mathcal{T}(W), \mathbf{x} \leftarrow \mathcal{T}_W(D), \alpha, \beta$
2: function RUN($q_{W,k}, \alpha, \beta, D$)
3:     $\epsilon \leftarrow$ TRANSLATE($q_{W,k}, \alpha, \beta$).$\epsilon^u$
4:     $(\tilde{x}_1, \ldots, \tilde{x}_L) = \mathbf{W}\mathbf{x} + Lap(b)^L$, where $b = k/\epsilon$
5:     $(i_1, \ldots, i_k) = \texttt{argmax}^k_{i=1,\ldots,L} \tilde{x}_i$
6:     return $(\{\phi_{i_1}, \ldots, \phi_{i_k}\}, \epsilon)$
7: end function
8: function TRANSLATE($q_{W,k}, \alpha, \beta$)
9:     return $\epsilon^u = \frac{2k \ln(L/(2\beta))}{\alpha}, \epsilon^l = \epsilon^u$
10: end function
```

Note that the privacy proof of the report-noisy-max algorithm does not work for releasing both the noisy count and the query number simultaneously. Hence we consider only releasing the bin identifiers in Algorithm 6. Moreover, the privacy cost of Algorithm 6 is independent of the workload $\|W\|_1$. On the other hand, the baseline LM (Algorithm 3) for answering TCQ queries is different from Algorithm 6. Algorithm 3 uses noise drawn from $Lap(\|W\|_1/\epsilon)$ to release noisy counts for all queries, and then picks the top-k as a post-processing step. Algorithm 3 allows the noisy counts to be shown to the data scientist without hurting the privacy cost. Hence, Algorithm 3 has a simpler privacy proof than Algorithm 6 and a privacy loss that depends on the workload. APEx supports both Algorithm 3 and Algorithm 6 as there is no clear winner between them when $k > 1$. APEx chooses the one with the least epsilon for a given accuracy bound.

3.4 Privacy Analysis

The privacy analyzer ensures that every sequence of queries answered by APEx results in a B-differentially private execution, where B is the data owner specified privacy budget. According to sequential composition [51], the privacy loss of a set of differentially private mechanisms (that use independent random coins) is the sum of the privacy losses of each of these mechanisms. Moreover, postprocessing the outputs of a differentially private algorithm does not degrade privacy [52].

The main tricky (and novel) part of the privacy proof (described in Section 3.4.1) arises due to the fact that 1) the ϵ parameter for a mechanism is chosen based on the scientist's query and accuracy requirement, which in turn are adaptively chosen by the data scientist based on previous queries and answers; and 2) some mechanisms may have an actual privacy loss that is dependent on the data. APEx accounts for privacy based on the actual privacy loss and not the worst case privacy loss (Algorithm 2, Line 12).

3.4.1 Overall Privacy Guarantee

We show the following guarantee: any sequence of interactions between the data scientist and APEx satisfies B-differential privacy, where B is the privacy budget specified by the data owner. In order to state this guarantee formally, we first need the notion of a *transcript of interaction* between APEx and the data scientist.

We define the transcript of interaction $\mathbb{T}$ as an alternating sequence of queries (with accuracy requirements) posed to APEx and answers returned by APEx. $\mathbb{T}$ encodes the scientist's view of the private database. More formally,

- The transcript $\mathbb{T}_i$ after i interactions is a sequence $[(q_1, \alpha_1, \beta_1), (\omega_1, \epsilon_1), \ldots, (q_i, \alpha_i, \beta_i), (\omega_i, \epsilon_i)]$, where (q_i, α_i, β_i) are queries with accuracy requirements, and ω_i is the answer returned by APEx and ϵ_i the actual privacy loss.
- Given $\mathbb{T}_{i-1}$, the data scientist chooses the next query $(q_{i+1}, \alpha_{i+1}, \beta_{i+1})$ adaptively. We model this using a (possibly randomized) algorithm $\mathbb{C}$ that maps a transcript $\mathbb{T}_{i-1}$ to (q_i, α_i, β_i); i.e., $\mathbb{C}(\mathbb{T}_{i-1}) = (q_i, \alpha_i, \beta_i)$. Note that the scientist's algorithm $\mathbb{C}$ does not access the private database D.
- Given (q_i, α_i, β_i), APEx select a subset of mechanisms $\mathcal{M}^*$ such that $\forall M \in \mathcal{M}^*$, $M.\textsc{translate}(q_i, \alpha_i, \beta_i).\epsilon^u \leq B - B_i$. Furthermore, if $\mathcal{M}^*$ is not empty, APEx chooses one mechanism $M_i \in \mathcal{M}^*$ deterministically (either based on ϵ^l or ϵ^u) to run. The selection of M_i is deterministic and independent of D.
- If APEx find no mechanism to run ($\mathcal{M}^* = \emptyset$), then the query is *declined* by APEx. In this case, $\omega_i = \perp$ and $\epsilon_i = 0$.
- If the APEx chosen algorithm M_i is LM, WCQ-SM, ICQ-SM or TCQ-LTM, $\epsilon_i = \epsilon_i^u$, where ϵ_i is the upperbound on the privacy loss returned by $M_i.\textsc{translate}$. For ICQ-MPM, the actual privacy loss can be smaller; i.e., $\epsilon_i \leq \epsilon_i^u$.
- Let $Pr[\mathbb{T}_i|D]$ denote the probability that the transcript of interaction is $\mathbb{T}_i$ given input database D. The probability is over the randomness in the scientist's choices $\mathbb{C}$ and the randomness in the mechanisms $M_1, \ldots, M_i$ executed by APEx.

Not all transcripts of interactions are realizable under APEx. Given a privacy budget B, the set of valid transcripts is defined as:

Definition 9 (Valid Transcripts)**.** *A transcript of interaction $\mathbb{T}_i$ is a* valid *APEx transcript generated by Algorithm 2 if given a privacy budget B the following conditions hold:*

- $B_{i-1} = \sum_{j=1}^{i-1} \epsilon_j \leq B$, *and*
- *Either* $\omega_i = \perp$, *or* $B_{i-1} + \epsilon_i^u \leq B$.

We are now ready to state the privacy guarantee:

Theorem 8 (*A*PEx Privacy Guarantee)**.** *Given a privacy budget B, any valid APEx transcript $\mathbb{T}_i$, and any pair of databases D, D' that differ in one row (i.e., $|D \setminus D' \cup D' \setminus D| = 1$), we have:*

1. $B_i = \sum_{j=1}^{i} \epsilon_i \leq B$, *and*
2. $Pr[\mathbb{T}_i|D] \leq e^{B_i} Pr[\mathbb{T}_i|D']$.

Proof. (1) directly follows from the definition of a valid transcript and these are the only kinds of transcripts a data scientist sees when interacting with *A*PEx.

(2) can be shown as follows using induction.

Base Case: When the transcript is empty, $Pr[\emptyset|D] \leq e^0 Pr[\emptyset|D']$.

Induction step: Now suppose for all $\mathbb{T}_{i-1}$ of that encode valid *A*PEx transcripts of length $i-1$, $Pr[\mathbb{T}_{i-1}|D] \leq e^{B_{i-1}} Pr[\mathbb{T}_{i-1}|D']$. Let $\mathbb{T}_i = \mathbb{T}_{i-1}||[(q_i, \alpha_i, \beta_i), (\omega_i, \epsilon_i)]$ be a valid *A*PEx transcript of length i. Then:

$$\begin{aligned} Pr[\mathbb{T}_i|D] &= Pr[\mathbb{T}_{i-1}|D] Pr[[(q_i, \alpha_i, \beta_i), (\omega_i, \epsilon_i)]|D, \mathbb{T}_{i-1}] \\ &= Pr[\mathbb{T}_{i-1}|D] Pr[\mathbb{C}(\mathbb{T}_{i-1}) = (q_i, \alpha_i, \beta_i)] Pr[M_i(D) = (\omega_i, \epsilon_i)] \end{aligned}$$

Note that the data scientist's choice of query q_i and accuracy requirement depends only on the transcript $\mathbb{T}_{i-1}$ and not the sensitive database, and thus incurs no privacy loss. Thus, it is enough to show that

$$Pr[M_i(D) = (\omega_i, \epsilon_i)] \leq e^{\epsilon_i} Pr[M_i(D') = (\omega_i, \epsilon_i)]$$

Case 1: When $\omega_i \neq \perp$ and M_i is LM, WCQ-SM, ICQ-SM, or TCQ-LTM, the mechanism satisfies ϵ_i^u-DP and $\epsilon_i = \epsilon_i^u$. Therefore, $Pr[M_i(D) = (\omega_i, \epsilon_i)] \leq e^{\epsilon_i} Pr[M_i(D') = (\omega_i, \epsilon_i)]$.

Case 2: When $\omega_i \neq \perp$ and M_i is ICQ-MPM, the mechanism satisfies ϵ_i^u-DP across all outputs. However, when either mechanism outputs (ω_i, ϵ_i), for $\epsilon_i < \epsilon_i^u$, we can show that

$Pr[M_i(D) = (\omega_i, \epsilon_i)] \leq e^{\epsilon_i} Pr[M_i(D') = (\omega_i, \epsilon_i)]$. In the case of ICQ-MPM, if the algorithm returns after i iterations of the loop, the noisy answer is generated by a DP algorithm with privacy loss $\epsilon_i = \frac{j}{m}\epsilon_i^u$.

Case 3: Finally, when $\omega_i = \perp$ (i.e., the query is declined), the decision to decline depends on ϵ_i^u of all mechanism applicable to the query (which is independent of the data) rather than ϵ_i (which could depend on the data in the case of ICQ-MPM). Therefore, $Pr[M_i(D) = (\omega_i, \epsilon_i)] = Pr[M_i(D') = (\omega_i, \epsilon_i)]$ for all D, D'. The proof would fail if the decision to deny a query depends on ϵ_i. $\square$

3.5 Query Benchmark Evaluation

Name	D	**Query workload** W	**Query output**
QW1	Adult	"capital gain"$\in [0, 50)$, "capital gain"$\in [50, 100)$, ...,"capital gain"$\in [4950, 5000)$	bin counts
QW2	Adult	"capital gain"$\in [0, 50)$, "capital gain"$\in [0, 100)$, ..., "capital gain"$\in [0, 5000)$	bin counts
QW3	NYTaxi	"trip distance"$\in [0, 0.1)$, "capital gain"$\in [0, 50)$, ...,"capital gain"$\in [0, 50)$	bin counts
QW4	NYTaxi	($0 \leq$"total amount"$< 1\wedge$ "passenger"$= 1$),.., ($9 \leq$"total amount"$< 10\wedge$ "passenger"$= 10$)	bin counts
QI1	Adult	"capital gain"< 50, "capital gain"< 100,..., "capital gain"< 5000	bin ids having counts $> 0.1\lvert D\rvert$
QI2	Adult	($0 \leq$"capital gain"< 100, "sex"='M'),...($4500 \leq$"capital gain"< 5000, "sex"='F')	bin ids having counts $> 0.1\lvert D\rvert$
QI3	NYTaxi	"fare amount"$\in [0, 0, 1)$, "fare amount"$\in [0.1, 2)$,..., "fare amount"$\in [9.9, 10)$	bin ids having counts $> 0.1\lvert D\rvert$
QI4	NYTaxi	"total amount"$\in [0, 0, 1)$, "total amount"$\in [0.1, 2)$,..., "total amount"$\in [9.9, 10)$	bin ids having counts $> 0.1\lvert D\rvert$
QT1	Adult	"age"$= 0$,"age"$= 1$,...,"age"$= 99$	top 10 bins with highest counts
QT2	Adult	100 predicates on different attributes, e.g. "age"$= 1$, "workclass"="private",...	top 10 bins with highest counts
QT3	NYTaxi	("PUID"=1 $\wedge$"DOID"=1), ("PUID"=1 $\wedge$ "DOID"=2),..,("PUID"=10 $\wedge$ "DOID"=10)	top 10 bins with highest counts
QT4	NYTaxi	100 predicates on different attributes, e.g. "pickup date"$= 1$, "passenger count"$= 1$,...	top 10 bins with highest counts

Table 3.1: Query benchmarks includes 3 types of exploration queries on 2 datasets.

In this section, we evaluate APEx on real datasets using a set of benchmark queries. We show that:

- APEx is able to effectively translate queries associated with accuracy bounds into differentially private mechanisms. These mechanisms accurately answer a wide variety of interesting data exploration queries with moderate to low privacy loss.
- The set of query benchmarks show that no single mechanism can dominate the rest and APEx picks the mechanism with the least privacy loss for all the queries.

3.5.1 Setup

Datasets. Our experiments use two real world datasets. The first data set Adult was extracted from 1994 US Census release [49]. This dataset includes 15 attributes (6 continuous and 9 categorical), such as "capital gain", "country", and a binary "label" indicating whether an individual earns more than 5000 or not, for a total of $32,561$ individuals. The second dataset, refereed as NYTaxi, includes $9,710,124$ NYC's yellow taxi trip records [5]. Each record consists of 17 attributes, such as categorical attributes (e.g., "pick-up-location"), and continuous attributes (e.g., "trip distance").

Query Benchmarks. We design 12 meaningful exploration queries on Adult and NYTaxi datasets, summarized in Table 3.1. These 12 queries cover the three types of exploration queries defined in § 3.1.1, QW1-4, QI1-4, and QT1-4 corresponds to WCQ, ICQ, and TCQ respectively. Queries with number 1 and 2 are for Adult, and with number 3 and 4 are for NYTaxi. The predicate workload W cover 1D histogram, 1D prefix, 2D histogram and count over multiple dimensions. We set $\beta = 0.0005$ and vary $\alpha \in \{0.02, 0.04, 0.08, 0.16, 0.32, 0.64\}$.

Metrics. For each query (q, α, β), *APEx* outputs (ϵ, ω) after running a differentially private mechanism, where ϵ is the actual privacy loss and ω is the noisy answer. The empirical error of a WCQ $q_W(D)$ is measured as $\|\omega - q_W(D)\|_\infty/|D|$, the scaled maximum error of the counts. The empirical errors of ICQ $q_{W,>c}(D)$ and TCQ $q_{W,k}(D)$ are measured as $\|\alpha\|_\infty/|D|$, the scaled maximum distance of mislabeled predicates.

Implementation Details. *APEx* is implemented using python-3.4, and is run on a machine with 64 cores and 256 GB memory. We run *APEx* with optimistic mode. For strategy mechanism, we choose H_2 strategy (a hierarchical set of counts [113, 114, 123]) for all queries. The code, data and evaluation metrics are open sourced on GitHub: `https://github.com/cgebest/APEx`.

3.5.2 *APEx* End-to-End Study

We run *APEx* for the 12 queries shown in Table 3.1 with different accuracy requirements from $0.01|D|$ to $0.64|D|$ and $\beta = 0.0005$. We show in Figure 3.2 that a line connects points (α, ϵ^u) where ϵ^u is the upper bound on the privacy loss for the mechanism chosen by *APEx* for the given α. For all the queries except QI2 and QI3, the mechanism chosen for each α incurs an actual privacy cost at $\epsilon = \epsilon^u$ and the only variation in the empirical error, so the corresponding $(\hat{\alpha}/|D|, \epsilon)$ of 10 runs is shown as boxplots. For QI2 and QI3, both the empirical error and the actual privacy cost $(\hat{\alpha}/|D|, \epsilon)$ vary across runs and hence are plotted as points in Figure 3.2.

The empirical error α is always bounded by the theoretical α for all the queries. The gap between the theoretical line and the actual boxplots/points are: 1) the analysis of the

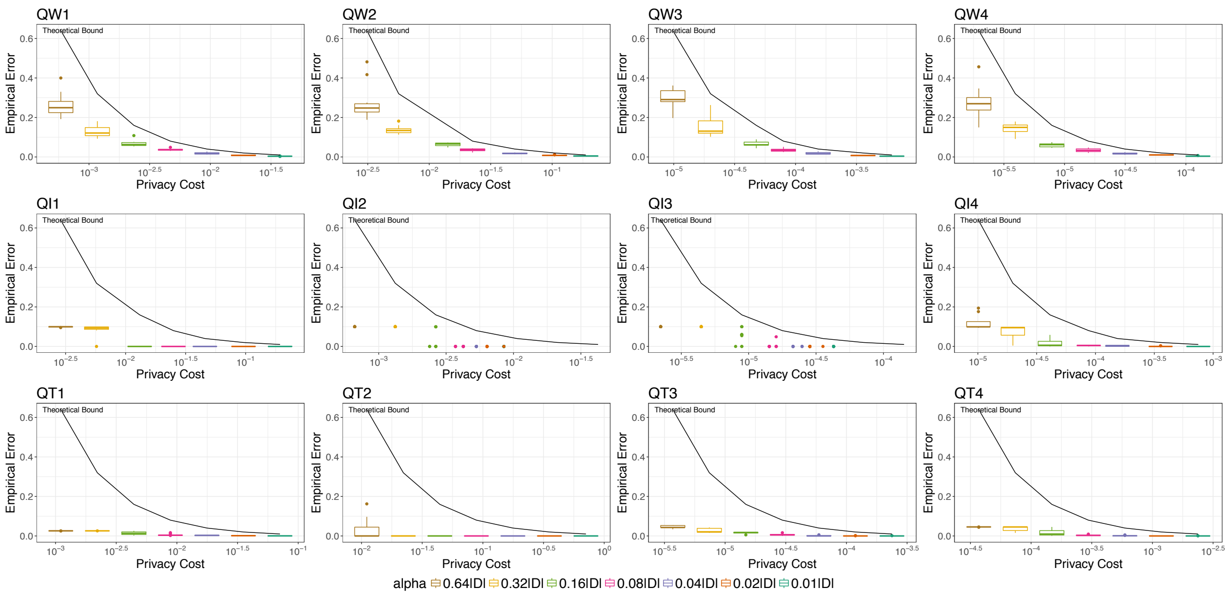


Figure 3.2: Privacy cost and empirical accuracy using optimal mechanism chosen by APEx (optimistic mode) on the 12 queries at default parameter setting with $\alpha \in \{0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64\}|D|$ and $\beta = 5 \times 10^{-4}$. On Adult data, all queries can be answered with empirical error < 0.1 with privacy budget < 0.1; on NYTaxi data, all queries can be answered with empirical error < 0.1 with privacy budget < 0.001. When accuracy requirement relaxes (i.e., α increases), the privacy cost decreases and the empirical accuracy decreases for all queries.

error is not tight due to the use of union bound; 2) for mechanism with data dependent translation (QI2 and QI3), the actual privacy cost is far from the upperbound ϵ^u resulting a left shift of the points from the theoretical line. The privacy cost for QW1 and QW2 for Adult dataset is in the range of $(10^{-4}, 10^{-1})$ for all α values. This privacy cost is 2-3 orders larger than the privacy cost for QW3 and QW4 on the NYTaxi dataset, because given the same ratio $\alpha/|D|$, the queries on NYTaxi has a larger α than Adult because of data size.

To further understand the relation between our accuracy requirement with commonly used error metric, we use F1 score to measure the similarity between the correct answer set and noisy answer set outputted from the mechanism. Figure 3.3 presents the F1 score of two queries: QI4 of an ICQ, and QT1 of a TCQ. In QT1, when α from $0.02|D|$ to $0.64|D|$, the median F1 score decreases from 0.9 to 0.15, which has a steeper gradient than the changes in Figure 3.2 and a closer trend with the theoretical α. In QI4, the F1 score is even more consistent with α and the empirical error shown in Figure 3.2. This shows that

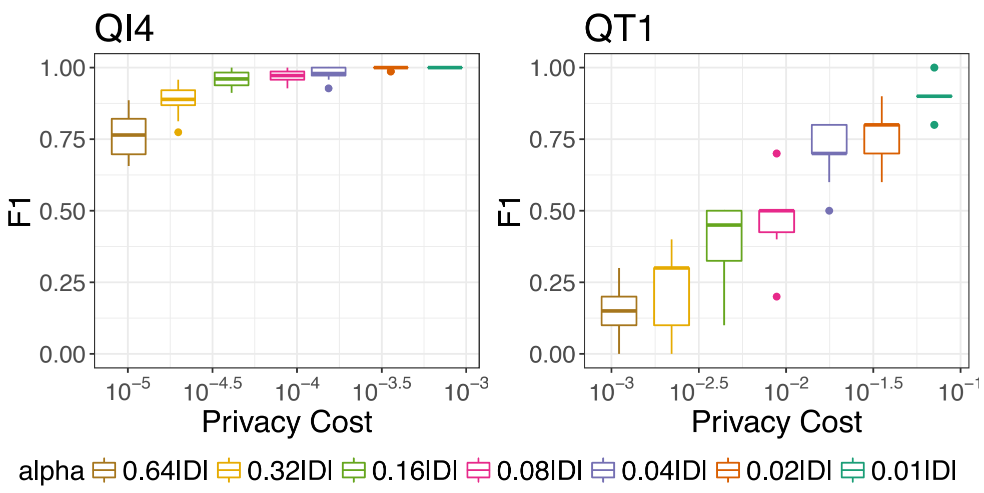


Figure 3.3: Privacy cost and empirical accuracy (F1 score) using optimal mechanism chosen by APEx (optimistic mode) on QI4 and QT1.

our (α, β) accuracy requirement is still a good indicator in data exploration process.

3.5.3 Optimal Mechanism Study

We run all the applicable mechanisms for the 12 queries from Table 3.1 at $\alpha \in \{0.02|D|, 0.08|D|\}$ and show the median of the actual privacy costs in Table 3.2. The privacy cost with the least value is in bold for the given query and accuracy. Indeed, APEx picks the mechanisms with these least privacy cost for all the 12 queries. APEx can save more than 90% of the privacy cost of the baseline translation (LM), such as QW2, QW3, QT2,QT4, and all the ICQ. In particular, the baseline mechanism LM is highly dependent on the sensitivity of the query. For example, the workload in QW2 (a cumulative histogram query) and QT2 (counts on many attributes) has a high sensitivity, so the cost of WCQ-LM is 20 times larger than WCQ-SM for QW2 at $\alpha = 0.08|D|$, and the cost of TCQ-LM is $760X$ more expensive than TCQ-LTM for QT2 at $\alpha = 0.02|D$, where WCQ-SM and TCQ-LTM are the optimal mechanisms chosen by APEx for QW2 and QT2 respectively.

The optimal mechanism chosen by APEx can change query parameters: workload size L, threshold c in ICQ, and k in TCQ. Figure 3.4 shows the effects of these parameters.

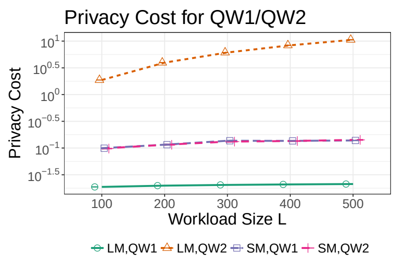
Vary Workload Size L. Figure 3.4a shows the privacy cost of the baseline mechanism WCQ-LM and the special mechanism WCQ-SM when varying workload size using QW1

Mechanism	**Query-α**							
	QW1-0.02$\|D\|$	QW1-0.08$\|D\|$	QW2-0.02$\|D\|$	QW2-0.08$\|D\|$	QW3-0.02$\|D\|$	QW3-0.08$\|D\|$	QW4-0.02$\|D\|$	QW4-0.08$\|D\|$
WCQ-LM	**0.01874**	**0.00469**	1.87430	0.46858	0.00629	0.00157	**0.00006**	**0.00002**
WCQ-SM	0.09880	0.02383	**0.10451**	**0.02251**	**0.00036**	**0.00008**	0.00033	0.00009
	QI1-0.02$\|D\|$	QI1-0.08$\|D\|$	QI2-0.02$\|D\|$	QI2-0.08$\|D\|$	QI3-0.02$\|D\|$	QI3-0.08$\|D\|$	QI4-0.02$\|D\|$	QI4-0.08$\|D\|$
ICQ-LM	1.76786	0.44197	0.01768	0.00442	0.00006	0.000015	0.00593	0.00148
ICQ-SM	**0.10271**	**0.02682**	0.10506	0.02517	0.00033	0.00008	**0.00034**	**0.00008**
ICQ-MPM	0.63644	0.31822	**0.00636**	**0.00371**	**0.00003**	**0.000014**	0.00640	0.00178
	QT1-0.02$\|D\|$	QT1-0.08$\|D\|$	QT2-0.02$\|D\|$	QT2-0.08$\|D\|$	QT3-0.02$\|D\|$	QT3-0.08$\|D\|$	QT4-0.02$\|D\|$	QT4-0.08$\|D\|$
TCQ-LM	**0.03536**	**0.00884**	266.24590	66.56148	**0.00012**	**0.00003**	1.40857	0.35214
TCQ-LTM	0.35358	0.08840	**0.35358**	**0.08840**	0.00119	0.00030	**0.00119**	**0.00030**

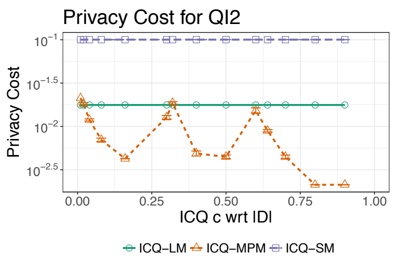
Table 3.2: Privacy cost using all applicable mechanisms on the 12 queries at $\alpha = \{0.02, 0.08\}|D|$ and $\beta = 5 \times 10^{-4}$. Median of 10 runs is reported for data-dependent mechanisms. It shows that (a) no single mechanism can always win or lose, (b) privacy cost of different mechanisms answering the same query, and privacy cost of same mechanism on different queries can be significantly different. Therefore, it is critical to use APEx for choosing optimal mechanisms.

and QW2 templates. WCQ-LM is highly dependent on the sensitivity of the query. Hence, the privacy cost of WCQ-LM on QW1 changes very slowly with the workload size as the sensitivity of the workload in QW1 is 1 while the privacy cost on QW2 is linear to workload size L because the the sensitivity of cumulative histogram is L. On the other hand, WCQ-SM incurs similar privacy cost on QW1 and QW2, since both queries share the same workload size and accuracy parameters. The privacy cost of WCQ-SM for QW1 and QW2 are similar as they are using the same H_2 strategy matrix. Their difference in Table 3.2 is mainly due to different runs of privacy cost estimation using MC simulation. However, their accuracy can be different, as QW2 with a cumulative histogram workload needs to add up $\log L$ number of noisy counts from the strategy matrix for one of its largest count. On the other hand, the histogram counts from QW1 requires only one noisy count. Similar findings have been observed on other query types and mechanisms; therefore, we omit to show other plots here.

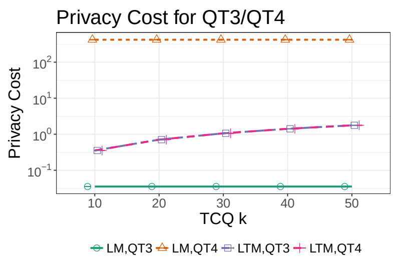
Vary counting threshold c. Figure 3.4b shows the actual privacy cost of mechanisms for QI2 with different thresholds. All the mechanisms for ICQ except ICQ-MPM, have a fixed privacy cost which is dependent of the data and the query (including c). However, we observe an interesting trend for the actual privacy cost used by ICQ-MPM as c increases. The smallest privacy cost which takes place after $c = 0.8$ is 1/10 of the upper bound of ICQ-MPM. The privacy cost of ICQ-MPM depends on the number of poking times before returning a output, which is related to the distance between the threshold and the true counts associated to the predicates. If the predicates are far from the threshold, the fewer



(a) Vary Workload Size



(b) Vary ICQ c



(c) Vary TCQ k

Figure 3.4: Privacy cost with query specific parameters: (a) Increasing workload size causes faster privacy cost increase fro WCQ-SM than WCQ-LM. (b) Varying ICQ threshold c affects privacy cost ICQ-MPM. The closer c relates to the true count, more attempts is needed, hence more privacy cost. (c) Increasing TCQ k leads to faster privacy cost increase for TCQ-TM than TCQ-LM.

number of poking is required and hence a smaller privacy budget is spent. On the other hand, if the true count is very close to the threshold, then it requires a small noise and hence all the budget to decide the label of this predicate with confidence. When $c = 0.01|D|$, 98% predicates are within the range $[c - \alpha, c + \alpha]$, and hence to confidently decide the label for all these predicates require more poking and hence a larger privacy cost. Consider the many predicates having counts close to $0.01|D|$, the cost of ICQ-MPM is high. As c increases to $0.10|D|$, all predicates have sufficient different counts as c, then 1 or 2 times of poking are sufficient. When c continue increases to $0.32|D|$, there is a single predicate that with true count (which is $0.3117|D|$) closer to c , it again requires more pokings to make a confident decision. A similar behavior is seen when c is around $0.6050|D|$.

Moreover, in the bad cases where c is close to true counts, the actual privacy cost of ICQ-MPM might be more expensive than the baseline ICQ-LM. For example, when $c = 0.01|D|$, ICQ-LM is better. This is a case where APEx under optimistic mode fails to choose ICQ-LM as the optimal mechanism.

Vary top threshold k. Figure 3.4c shows the sensitivity of privacy cost of TCQ-LM and TCQ-TM with respect to varying k, using QT3 and QT4 as examples. The privacy cost of TCQ-LM is independent of k as shown in Figure 3.4c, but the cost of TCQ-LTM increases linearly with k. The privacy cost of TCQ-LTM for both QT3 and QT4 have the same privacy cost, because its cost only depends on k and the accuracy parameters, but the privacy cost of LTM for QT3 and QT4 are very different because the HD QT4 has larger sensitivity.

In summary, we see that the optimal mechanism with the least privacy cost changes among queries and even the same query with different parameters. This shows there is a great need for systems like APEx to provide translation and identify optimal mechanisms.

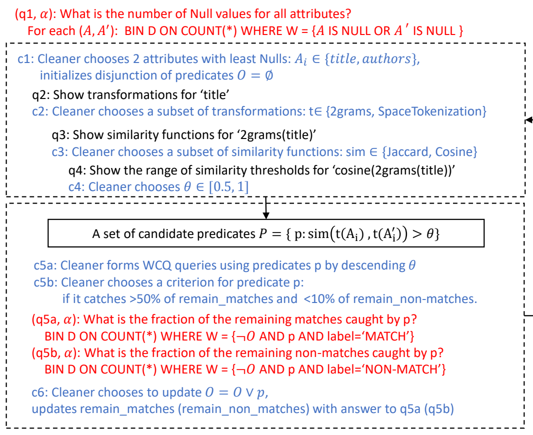
3.6 Case Study

In this section, we design an application benchmark based on entity resolution to show that 1) we can express real data exploration workflow using our exploration language; 2) APEx allows entity resolution workflow to be conducted with high accuracy and strong privacy guarantee; and 3) APEx allows trade-off between privacy and final task quality for a given exploration task.

3.6.1 Case Study Setting

Entity Resolution (ER) is an application of identifying table records that refer to the same real-world object. In this case, we use the `citations` [48] dataset to perform exploration tasks. Each row in the table is a pair of citation records, with a binary label indicating whether they are duplicates or not. All the citation records share the same schema, which consists of 3 text attributes of title, authors and venue, and one integer attribute of publication year. A training set D of size 4000 is sampled from `citations` such that every record appears at most once. Two exploration tasks for entity resolution on D are considered: *blocking* and *matching*. These two tasks are achieved by learning a boolean formula P (e.g. in DNF) over *similarity predicates*. We express a similarity predicate p as a tuple $(A, t, sim, \theta) \in attr(R) \times \mathbb{T} \times \mathbb{S} \times \Theta$, where $A \in attr(R)$ is an attribute in the schema $t \in \mathbb{T}$ is a transformation of the attribute value, and $sim \in \mathbb{S}$ is a similarity function that usually takes a real value often restricted to $[0, 1]$, and $\theta \in \Theta$ is a threshold. Given a pair of records (r_1, r_2), a similarity predicate p returns either 'True' or 'False' with semantics: $p(r_1, r_2) \equiv (sim(t(r_1.A), t(r_2.A)) > \theta)$.

In this case study, the exploration task for blocking is to find a boolean formula P_b that identifies a small set of candidate matching pairs that cover most of the true matches, known as high *recall*, with a small blocking cost (a small fraction of pairs in the data that return true for P_b). The exploration task for matching is to identify a boolean formula P_m that identifies matching records that achieves high *recall* and *precision* on D_t. Precision measures whether P_m classifies true non-matches as matches, and recall measures the fraction of true matches that are captured by P_m. The quality of this task is measured by $F_1^{P_m}$, the harmonic mean of precision and recall.

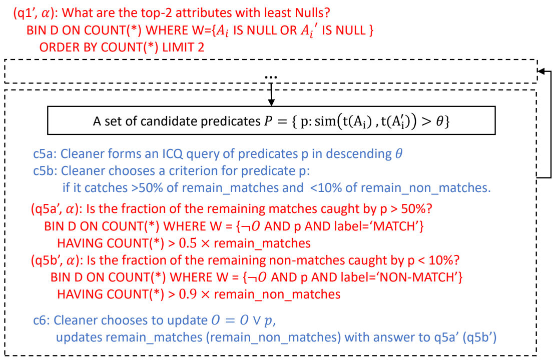

(q1, α): What is the number of Null values for all attributes?
For each (A, A'): BIN D ON COUNT(*) WHERE W = {A IS NULL OR A' IS NULL }

c1: Cleaner chooses 2 attributes with least Nulls: $A_i \in \{title, authors\}$,
initializes disjunction of predicates $O = \emptyset$
q2: Show transformations for 'title'
c2: Cleaner chooses a subset of transformations: t∈ {2grams, SpaceTokenization}
q3: Show similarity functions for '2grams(title)'
c3: Cleaner chooses a subset of similarity functions: sim ∈ {Jaccard, Cosine}
q4: Show the range of similarity thresholds for 'cosine(2grams(title))'
c4: Cleaner chooses $\theta \in [0.5, 1]$

A set of candidate predicates $P = \{\, p: \mathrm{sim}(\mathrm{t}(A_i), \mathrm{t}(A_i')) > \theta \}$

c5a: Cleaner forms WCQ queries using predicates p by descending θ
c5b: Cleaner chooses a criterion for predicate p:
if it catches >50% of remain_matches and <10% of remain_non-matches.
(q5a, α): What is the fraction of the remaining matches caught by p?
BIN D ON COUNT(*) WHERE W = {$\neg O$ AND p AND label='MATCH'}
(q5b, α): What is the fraction of the remaining non-matches caught by p?
BIN D ON COUNT(*) WHERE W = {$\neg O$ AND p AND label='NON-MATCH'}
c6: Cleaner chooses to update $O = O \vee p$,
updates remain_matches (remain_non_matches) with answer to q5a (q5b)

(a) Strategy instance 1 for blocking


(q1', α): What are the top-2 attributes with least Nulls?
BIN D ON COUNT(*) WHERE W={A_i IS NULL OR A_i' IS NULL }
ORDER BY COUNT(*) LIMIT 2

...

A set of candidate predicates $P = \{\, p: \mathrm{sim}(\mathrm{t}(A_i), \mathrm{t}(A_i')) > \theta \}$

c5a: Cleaner forms an ICQ query of predicates p in descending θ
c5b: Cleaner chooses a criterion for predicate p:
if it catches >50% of remain_matches and <10% of remain_non_matches.
(q5a', α): Is the fraction of the remaining matches caught by p > 50%?
BIN D ON COUNT(*) WHERE W = {$\neg O$ AND p AND label='MATCH'}
HAVING COUNT(*) > 0.5 × remain_matches
(q5b', α): Is the fraction of the remaining non-matches caught by p < 10%?
BIN D ON COUNT(*) WHERE W = {$\neg O$ AND p AND label='NON-MATCH'}
HAVING COUNT(*) > 0.9 × remain_non_matches
c6: Cleaner chooses to update $O = O \vee p$,
updates remain_matches (remain_non_matches) with answer to q5a' (q5b')

(b) Strategy instance 2 for blocking

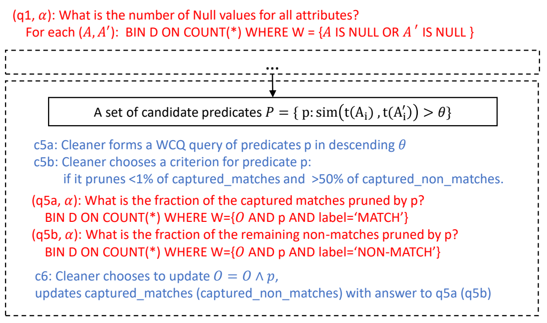
Figure 3.5: Two strategies for blocking.

Using exploration queries supported by APEx, we can express two exploration strategies for each task: BS1 (MS1) using WCQ only to complete blocking (matching) task, and BS2 (MS2) using ICQ/TCQ to conduct the blocking (matching) task. Each strategy generates a sequence of exploration queries which interact with APEx and constructs a boolean formula for the corresponding task. For each strategy, we randomly sample a concrete cleaner from our cleaner model and report mean and quartiles of its cleaning quality over 100 runs.

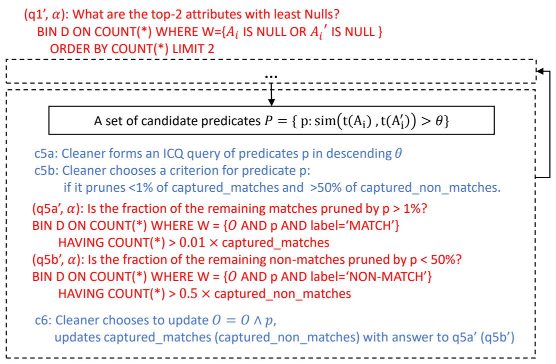
3.6.2 The ER Model

The data scientist typically narrows down the choices through a sequence of actions, consisting of issuing queries and making choices based on the answer of issued query. We use a *strategy* to denote a class of actions that use the same set of queries but make different choices. Figure 3.5 shows the strategies for the blocking task. These two strategies are different as they use different query types, though they share the same criteria of decision choices. In particular, the queries $q1, q5$ in the strategy shown in Figure 3.5a are WCQ, while the queries $q1', q5'$ in Figure 3.5b are TCQ and ICQ respectively. This case exemplifies how a cleaning engineer constructs a single path (disjunction) of predicates to form a blocking function, though a real function can be more complex [58]. Similarly, Figure 3.6 illustrates two matching strategies using WCQ (Figure 3.6a) and ICQ/TCQ (Figure 3.6b), respectively, where the matching function is formed as a conjunction of predicates.

The ER model encodes the space of all the parameters involved in the data scientist's


(q1, α): What is the number of Null values for all attributes?
For each (A, A'): BIN D ON COUNT(*) WHERE W = {A IS NULL OR A' IS NULL }
...
A set of candidate predicates $P = \{ \text{p}: \text{sim}(\text{t}(\text{A}_\text{i}), \text{t}(\text{A}'_\text{i})) > \theta\}$
c5a: Cleaner forms a WCQ query of predicates p in descending θ
c5b: Cleaner chooses a criterion for predicate p:
if it prunes <1% of captured_matches and >50% of captured_non_matches.
(q5a, α): What is the fraction of the captured matches pruned by p?
BIN D ON COUNT(*) WHERE W={O AND p AND label='MATCH'}
(q5b, α): What is the fraction of the remaining non-matches pruned by p?
BIN D ON COUNT(*) WHERE W={O AND p AND label='NON-MATCH'}
c6: Cleaner chooses to update $O = O \wedge p$,
updates captured_matches (captured_non_matches) with answer to q5a (q5b)

(a) Strategy instance 1 for matching (MS1)


(q1', α): What are the top-2 attributes with least Nulls?
BIN D ON COUNT(*) WHERE W={A_i IS NULL OR A_i' IS NULL }
ORDER BY COUNT(*) LIMIT 2
...
A set of candidate predicates $P = \{ \text{p}: \text{sim}(\text{t}(\text{A}_\text{i}), \text{t}(\text{A}'_\text{i})) > \theta\}$
c5a: Cleaner forms an ICQ query of predicates p in descending θ
c5b: Cleaner chooses a criterion for predicate p:
if it prunes <1% of captured_matches and >50% of captured_non_matches.
(q5a', α): Is the fraction of the remaining matches pruned by p > 1%?
BIN D ON COUNT(*) WHERE W = {O AND p AND label='MATCH'}
HAVING COUNT(*) > 0.01 × captured_matches
(q5b', α): Is the fraction of the remaining non-matches pruned by p < 50%?
BIN D ON COUNT(*) WHERE W = {O AND p AND label='NON-MATCH'}
HAVING COUNT(*) > 0.5 × captured_non_matches
c6: Cleaner chooses to update $O = O \wedge p$,
updates captured_matches (captured_non_matches) with answer to q5a' (q5b')

(b) Strategy instance 2 for matching (MS2)

Figure 3.6: Two strategies for matching.

decisions $c1$-$c6$. Table 3.3 summarizes the space of all the parameters for $c1$-$c6$ in blocking strategy 1. From $c1$ to $c4$, the program chooses (i) a subset of attributes x_1 of size ranging from 2 up to the total number of attributes $|attr(R)|$, (ii) a subset of transformations x_2 from $\mathbb{T} = \{2grams, 3grams, SpaceTokenization\}$, (iii) a subset of similarity functions x_3 from $\mathbb{S} = \{Edit, SmithWater, Jaro, Cosine, Jaccard, Overlap, Diff\}$, and (iv) x_6 thresholds from the range of $[x_4, x_5]$, where $x_4 \in (0, 0.5)$, $x_5 \in (0.5, 1)$, and $x_6 \in \{2, 3, 4, 5, 6\}$. The cross product of these choices form a set of predicates P, and $c5a$ picks an ordering x_7, one of the permutation of P. In $c5b$, the model sets the criterion for pruning or keeping a predicate p from the top list of P. In particular, the model sets x_8 and x_9 as the minimum fraction of the remaining matches caught and the maximum fraction of the remaining non-matches caught by $p \vee O$ respectively, where $x_8 \in [0.2, 0.5]$ and $x_9 \in [0.1, 0.2]$. These values are reset as $x_8 = x_8/x_{10}$ and $x_9 = x_9 x_{10}$ where $x_{10} \in \{2, 3\}$ if all predicates have been checked but $O = \emptyset$. In $c6$, the model considers three possible styles of cleaners on trusting the noisy answers: *neutral* style corresponds to trust the noisy answers; for *optimistic* (*pessimistic*) style, the data scientist trusts the values by adding (subtracting) $\alpha/5$ to (from) the noisy answers. If these criterion are met and the blocking cost over D_t is less than a fixed cutoff threshold (e.g., a hardware constraint, we set 550). An instance of all variables $\mathcal{C} = (x_1, \ldots, x_{11})$ in Table 3.3 forms a concrete cleaner. The model for other strategies is similarly constructed.

3.6.3 End-to-End Task Evaluation

We report the end-to-end performance of $\mathcal{A}$PEx on the four exploration strategies.

Table 3.3: A cleaner model for blocking strategy 1.

A cleaner model $\mathcal{C} = \{x_1, \ldots, x_{11}\}$ for c1-c6 in Figure 3.5a	
$c1$	Choose x_1, an ordered subset of attributes with least Nulls, where $x_1 \in \{2, 3, \lvert attr(R)\rvert\}$
$c2$	Choose x_2, an ordered subset of transformations from $\mathbb{T}$, where $\lvert x_2\rvert \in \{1, 2, 3\}$
$c3$	Choose x_3, an ordered subset of similarity functions from $\mathbb{S}$, where $\lvert x_3\rvert \in \{2, 3, 4, 5, 6\}$
$c4$	Choose a lower bounds x_4 and a upper bound x_5 of threshold range, where $x_5 \in (0.5, 1)$, $x_4 \in (0, 0.5)$, and evenly choose x_6 thresholds in order of either ASC or DSC, and $x_6 \in \{2, 3, 4, 5, 6\}$
$c5a$	Predicate p is sequentially selected based the order of $x_7 \in \text{Permute}(x_1 \times x_2 \times x_3 \times x_6)$
$c5b$	Chooses a criterion for predicate p: if it catches $< x_8$ fraction of the remaining matches and $> x_9$ fraction of remaining non-matches, where $x_8 \in [0.2, 0.5]$ and $x_9 \in [0.1, 0.2]$. Reset $x_8 = x_8/x_{10}$ and $x_9 = x_{10}x_9$, $x_{10} \in \{2, 3\}$ if all queries have been asked but $O = \emptyset$.
$c6$	Choose method $x_{11} \in \{\text{neutral, optimistic, pessimistic}\}$ to take tolerance into account. If conditions are met, and the blocking cost is less than cutoff threshold, add p to the output O and remove it from P.

Vary Privacy Constraint. Given an exploration strategy from BS1, BS2, MS1, MS2 on training data D, a sequence of queries is generated based on its interaction with APEx until the privacy loss exceeds the privacy constraint B specified by the data owner. The accuracy requirement for this set of experiments is fixed to be $\alpha = 0.008|D_t|, \beta = 0.0005$. The privacy constraint varies from 0.1 to 2. Each exploration strategy is repeated 100 times under a given privacy constraint and we report the output quality of these 100 runs at different privacy constraint.

Figure 3.7 shows the exploration quality (recall for blocking task and F1 for matching task) of 100 runs of each exploration strategy at $B = \{0.1, 0.2, 0.5, 1, 1.5, 2\}$. We observe that the expected task quality (median and variance) improves as the budget constraint increases and gets stable after, for example reaching $B \geq 1$ for MS1. Since we fixed α, the privacy loss for each exploration query is also fixed. Thus, B directly controls the number of queries APEx answers before halting. For small $B < 0.2$, only a few queries are answered and the cleaning quality is close to random guessing. As B increases, more queries can be learned about the dataset. After B reaches a certain value around 1.0, APEx can answer sufficient number of queries; therefore obtaining high accuracy. For MS2, when it reaches

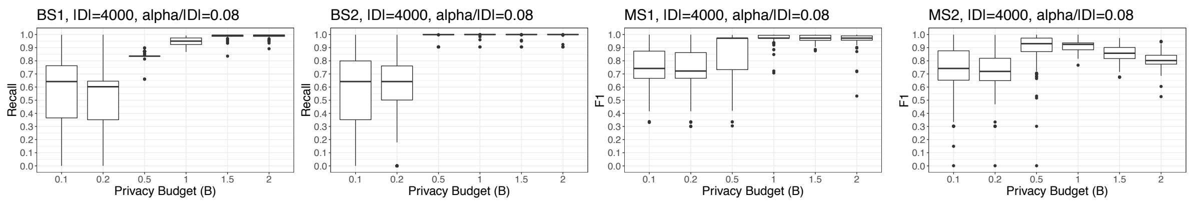


Figure 3.7: Performance of $\mathcal{A}$PEx for blocking (BS1, BS2) and matching (MS1, MS2) tasks with increasing privacy budget B at fixed $\alpha = 0.08|D|$: the expected task quality improves with smaller variance as the budget constraint increases and gets stable. Fixing α fixes privacy cost per operation. Thus increasing B increases the number of queries answered.

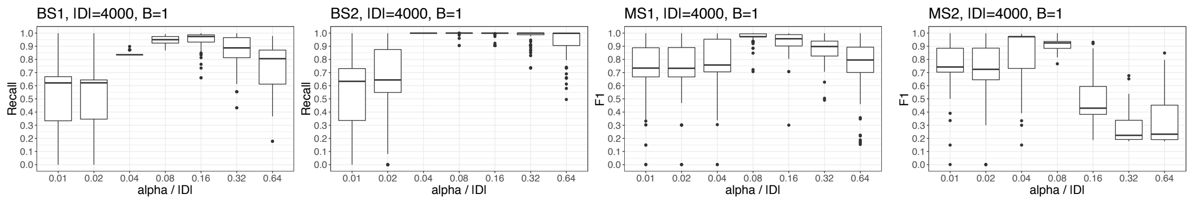


Figure 3.8: Performance of $\mathcal{A}$PEx for blocking (BS1, BS2) and matching (MS1, MS2) at fixed privacy budget $B = 1$ with increasing α from $0.01|D_t|$ to $0.64|D_t|$. There exists an optimal α to achieve highest quality at a given privacy constraint. Increasing α decreases privacy cost per operation. Thus for a fixed budget this increases the number of queries. However, many queries each with a low privacy budget is not good for end-to-end accuracy.

good F1 score at $B = 1$, more noisy answers can mislead the MS2 strategy to add more predicates to the blocking conjunction function, and decrease the quality.

The privacy cost used by each ICQ and TCQ is generally less than what has spent on the corresponding WCQ, as less information is shown to the data scientist. Given the same privacy budget, e.g. when $B = 0.5$, more queries can be answered in BS2 than BS1, results in a 25% better recall. We observe the same trend in MS1 and MS2. This shows that it is important for $\mathcal{A}$PEx to support translation for different types of queries for better end-to-end accuracy.

Vary Accuracy Requirement. This section shows the task quality of each exploration strategy at different accuracy requirements under a fixed privacy constraint $B = 1.0$. For each $\alpha \in \{0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64\}|D|$, the exploration strategy interacts with $\mathcal{A}$PEx until the privacy loss exceeds $B = 1.0$. Each exploration is repeated 100 times and we report the quality of the constructed boolean formula of these 100 runs. As shown in Figure 3.8, the quality of the four exploration strategies all improves first as accuracy requirement relaxes and then degrades again under $B = 1.0$. This is because when there

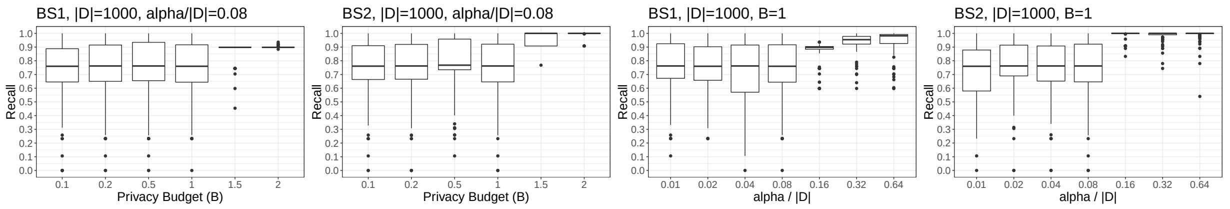


Figure 3.9: Performance of APEx for blocking (BS1, BS2) tasks at $|D| = 1000$. Compared with Figure 3.7 where $\alpha = 0.08|D|$, the privacy budget that needs to achieve good recall increases when data size is smaller. Compared with Figure 3.8 where $B = 1$, the optimal α actually increases.

is a privacy constraint, APEx only allows a limited number of queries to be answered. Given this fixed privacy budget $B = 1$ and a fixed accuracy requirement α, the number of queries can be answered is bounded and related to α. When α first increases from $0.01|D|$ to $0.08|D|$, more queries can be answered and give more information on which predicates to choose. However, as α keeps growing, the answers get noisier and misleading, resulting in the drop in quality, even though more questions can be answered.

Vary Data Size. We also experimented with the same strategies with a different data size $|D| = 1000$, using blocking strategies shown in Figure 3.9 as examples to study the affects of data size. Comparing with Figure 3.7 where α is fixed at $0.08|D|$, the privacy cost to achieve optimal recall is larger when $|D|$ is smaller. BS1 and BS2 can achieve very good recall when $|D| = 4000$ given a privacy budget $B = 1$, but they requires at least $B = 1.5$ to reach similar quality when $|D| = 1000$. On the other hand, comparing with Figure 3.8 where the privacy budget is fixed $B = 1$, the optimal α for smaller $|D|$ is very close. BS1 and BS2 reach the highest recall at about $\alpha = 0.16 \cdot 1000 = 160$ which is close to the optimal $\alpha = 0.8 \cdot 4000 = 320$ when $|D| = 4000$. Thus suggest an interesting direction for future work: choosing the optimal α for different queries in an exploration process.

3.7 Discussion and Conclusion

3.7.1 Other Aggregation Functions

Besides supporting linear counting queries, our algorithms in APEx can be easily extended to support SUM() queries. Queries for MEDIAN() (and percentile) can be supported by first querying the CDF (using a WCQ), and finding the median from that. GROUPBY can be expressed as a sequence of two queries: first query for values of an attribute with

COUNT(*) > 0 (using ICQ), and then query the noisy counts for these bins (using WCQ). Similarly, if the aggregated function $f()$ in HAVING differs from the aggregated function $g()$, APEx can express it as a sequence of two queries: first query for bins having $f() > c$ (using ICQ), and then apply $g()$ on these bins (using WCQ).

However, no differentially private algorithms can accurately report MAX() and MIN(). Non-linear queries such as AVG() and STD() can have very sensitive error bounds to noise when computed on a small number of rows (like the measures discussed in § 3.1.2). Hence, supporting non-linear queries would need new translation mechanisms. Moreover, APEx can support queries with foreign key joins (that do not amplify the contribution of one record), but designing accurate differentially private algorithms for queries with general joins is still an active area of research.

3.7.2 Conclusion

We proposed APEx, a framework that allows the data scientist to interact with sensitive data while ensuring that their interactions satisfy differential privacy. Using experiments with query benchmarks and entity resolution application, we established that APEx allows high exploration quality with a reasonable privacy loss.

APEx opens many interesting future research directions. First, more functionalities can be added to APEx: a) a recommender which predicts the subsequent interesting queries and advises the privacy cost of these queries to the data scientist; b) an inferencer which uses historical answers to reduce the privacy cost of a new incoming query; and c) a sampler which incorporate approximate query processing to have 3-way trade-off between efficiency, accuracy, and privacy. More translation algorithms from accuracy to privacy, especially for data-dependent mechanisms and non-linear queries can be implemented in APEx to further save privacy budget throughout the exploration. We show the exploration queries in APEx to support entity resolution tasks. These exploration queries can be extended to support other exploration tasks, e.g., schema matching, feature engineering, tuning machine learning workloads. This would also require extending our system to handle relations with multiple tables, constraints like functional dependencies. APEx turns the differentially private algorithm design problem on its head – it minimizes privacy loss given an accuracy constraint. This new formulation has applications on sensitive data exploration and can trigger an entirely new line of research.

Chapter 4

Kamino: Constraint-Aware Differentially Private Data Synthesis

4.1 Problem Statement and Solution Overview

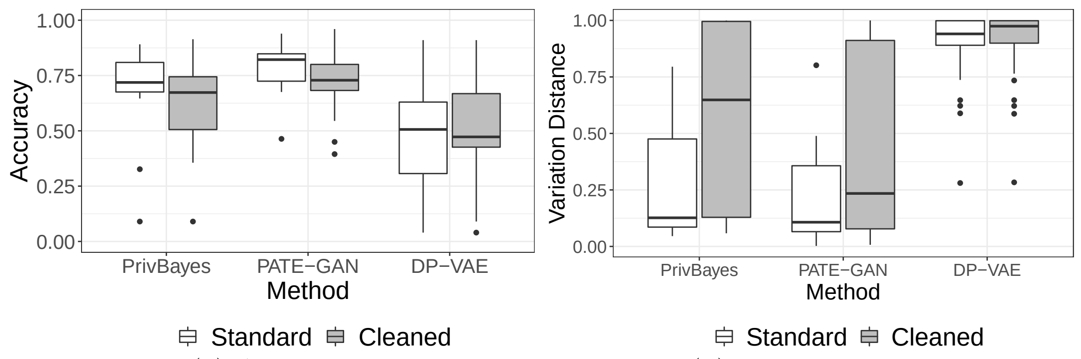
To solve the shortcomings of the current differentially private data synthesis approaches mentioned in § 1.2, we state our problem definition and provide a high-level description of our approach.

4.1.1 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.



(a) Accuracy (b) 2-way marginal

Figure 4.1: 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 4.1a represents the testing accuracy for one target attribute. Each point in Figure 4.1b represents the total variation distance between the true and synthetic Adult. More details are in § 4.5.

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. The semantic privacy results by Ganta et al. [67, 98] (§ 2.1.3) 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.

4.1.2 A Naïve Attempt

We use the following example to show that 1) the synthetic data generated by state-of-the-art private data synthesis methods has missing DCs; and 2) a post-cleaning step with the state-of-the-art data cleaning technique [148] on the synthetic data can improve the data consistency, but it is at the cost of the usefulness of the synthetic for other applications.

Example 6: Consider the Adult dataset [49] consisting of 15 attributes with denial constraints [88], 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 [174], PATE-GAN [95], and DP-VAE [38] have up to 32% of the tuple pairs failing these constraints (Table 4.2).

However, naïvly repairing the incorrect structure constraints in the synthetic data can compromise the usefulness. We applied state-of-the-art data cleaning method [148] 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 4.1 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. □

4.1.3 Methodology Overview

Recall from § 2.1.2, 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: 1) privately learn the unknown parameters in the probabilistic database model with samples from the true data; and 2) 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 [150, 156], and approximate methods such as gradient descent may not converge to a global optimum [151], 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 [30, 55, 65, 159] 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. (2.1) into a chain of conditional probabilities based on two sequences 1) a sequence of tuple ids; and 2) 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

$$\begin{aligned}|V(\phi, D)| &= |V(\phi, t_1)| + |V(\phi, t_2 \mid D_{:2})| + \cdots + |V(\phi, t_n \mid D_{:n})| \\ &= \sum_{i=1}^{n} |V(\phi, t_i \mid D_{:i})| \qquad (4.1)\end{aligned}$$

This allows us to decompose Eqn. (2.1) as

$$\begin{aligned}\Pr(D) &\propto \left(\prod_{i=1}^{n} \Pr(t_i)\right) \times \exp\left(-\sum_{\phi\in\Phi} w_\phi \sum_{i=1}^{n} |V(\phi, t_i \mid D_{:i})|\right) \\ &= \prod_{i=1}^{n} \left[\Pr(t_i) \times \exp\left(-\sum_{\phi\in\Phi} w_\phi \times |V(\phi, t_i \mid D_{:i})|\right)\right] \qquad (4.2)\end{aligned}$$

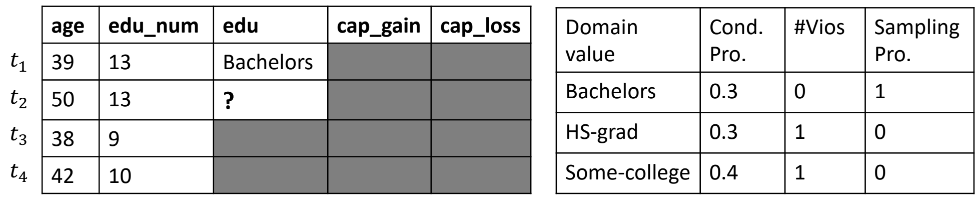
Next, we define a schema sequence S as an ordered list of all attributes in the schema. Similarly, let $S_{:j}$ represent all prefix attributes of the jth attribute in S and $S_{:1} = \emptyset$ for the purpose of uniform representation. This schema sequence allows us further decompose the set of violations. 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 7: Continue with Example 4, 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. (4.2) as follows:

$$\begin{aligned}\sum_{\phi\in\Phi} w_\phi \times |V(\phi, t_i \mid D_{:i})| &= \sum_{j=1}^{k} \sum_{\phi\in\Phi_{A_j}} w_\phi \times |V(\phi, t_i \mid D_{:i})| \\ &= \sum_{j=1}^{k} \sum_{\phi\in\Phi_{A_j}} w_\phi \times |V(\phi, t_i[S_{:j+1}] \mid D_{:i}[S_{:j+1}])| \qquad (4.3)\end{aligned}$$

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.



	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 4.2: Sampling values in an instance (Example 8).

Finally, we have the database probability in Eqn. (4.2) expressed as

$$\Pr(D) \propto \prod_{j=1}^{k}\prod_{i=1}^{n}\Big[\Pr(t_i[A_j] \mid t_i[S_{:j+1}])\times \\ \exp(-\sum_{\phi\in\Phi_{A_j}} w_\phi \times |V(\phi, t_i[S_{:j+1}] \mid D_{:i}[S_{:j+1}])|)\Big] \tag{4.4}$$

Eqn. (4.4) 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 [42], 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 8: Continue with Examples 4 and 7. 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 *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 4.2), 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 [134] 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 7 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 10, 8 and 9) when the weights of DCs are given in § 4.2, and then explain how to learn the DC weights (Algorithm 11) in § 4.3. Last, privacy analysis and parameter search (Algorithm 12) are explained in § 4.4.

Our system assumes the inputs are static, since we rely on the database instance to learn the generative process (i.e., Algorithms 8, 9 and 11), 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 [47] for dynamically growing databases for better utility.

4.2 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.

Algorithm 7 Constraint-aware differentially private data synthesis

```
Input: Private instance $D^*$, schema $R$, domain $\mathcal{D}$
Input: DCs $\Phi$, privacy budget $(\epsilon, \delta)$
1: procedure KAMINO($D^*, R, \mathcal{D}, \Phi, \epsilon, \delta$)
2:     $S \leftarrow$ SEQUENCING($R, \mathcal{D}, \Phi$)                    ▷ Algorithm 10
3:     $\Psi \leftarrow$ SEARCHDPARAS($\epsilon, \delta, \mathcal{D}, S$)            ▷ Algorithm 12
4:     $M \leftarrow$ TRAINMODEL($D^*, S, \mathcal{D}, \Psi$)               ▷ Algorithm 8
5:     $W \leftarrow$ LEARNWEIGHT($D^*, \Phi, S, M, \Psi$)            ▷ Algorithm 11
6:     $D' \leftarrow$ SYNTHESIZE($S, M, \Phi, \mathcal{D}, W$)            ▷ Algorithm 9
7:     return $D'$
8: end procedure
```

4.2.1 Private Learning of Tuple Probability

Recall Equ. (2.2) that, given a schema sequence $S = [A_1, A_2, \ldots, A_k]$, the tuple probability becomes $\Pr[t] = \Pr(t[A_1]) \cdot \prod_{j=2}^{k} \Pr(t[A_j] \mid t[A_1, \ldots, A_{j-1}])$. 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 [56] 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 $\Pr(t[A_j] \mid t[A_1, \ldots, A_{j-1}])$, 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 (§ 2.1.2). The training of each discriminative sub-model on the samples from the true data is optimized and privatized using DPSGD [11, 22, 155, 164].

Algorithm 8 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 § 4.2.3), 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 § 4.4 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

Algorithm 8 Probabilistic data model training

```
Input: $D^*, \mathcal{D}, S$                         ▷ True instance, domain, schema sequence
Input: $n, k, \eta, q$                          ▷ cardinality, dimensions, lr, quantization
Input: $\sigma_g, \sigma_d$                                          ▷ Noise scales in $\Psi$
Input: $C, T, b$                    ▷ $L_2$ norm clip/#iterations/batch size in $\Psi$
 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 $M_{X,y}$                             ▷ [165]
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
```

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). The final output from Algorithm 8 is the probabilistic data model M, which will be used to sample tuple values in § 4.2.2.

Algorithm 8 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, Algorithm 8 satisfies differential privacy. We will analyze the privacy cost in § 4.4. 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 will evaluate this trade-off in § 4.5.3.

4.2.2 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 9.

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 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.

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 $\mathcal{O}(dn)$ (for an unary DC) or $\mathcal{O}(dn^2)$ (for a binary DC). This can be optimized by exploiting the property of hard functional dependencies, and we will evaluate one optimization in § 4.5.3. In addition, when $m > 0$ for MCMC, the sampling algorithm has an additional cost of

Algorithm 9 Constraint-aware database instance sampling

```
Input: $S, M, \Phi, \mathcal{D}$        ▷ Schema sequence, data model, DCs, domain
Input: $W, L, N$        ▷ Weight vector (Alg. 11), 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 $\phi$ 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
```

$\mathcal{O}(mkd+|\Phi|dnm)$. The overall complexity of constraint-aware sampling is $\mathcal{O}(nkd+|\Phi|dn^2+mkd+|\Phi|dnm)$.

We expect the synthetic database instance has similar number of violations as the truth. We provides a theoretical analysis at the end (§ 4.6.1).

4.2.3 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 [44, 171] 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 10 describes the process of finding a schema sequence S. For the list of FDs $\Sigma = [X_1 \rightarrow$

Algorithm 10 Constraint-aware attribute sequencing

```
Input: $R, \mathcal{D}, \Phi$                    ▷ Input schema, domain, and DCs
1: procedure SEQUENCING($R, \mathcal{D}, \Phi$)
2:     $\Sigma \leftarrow$ FDs from $\Phi$ 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 its 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
```

$Y_1, \ldots, X_m \rightarrow Y_m]$, 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 $\mathcal{O}(k|\Sigma| + \log k)$, 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 10 returns a sequence based on the domain size. Following this sequence, each discriminative sub-model (§ 4.2.1) 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.

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 10 on the BR2000 dataset [174] 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 [43] with 30k tuples has one *zip* attribute

Algorithm 11 Learning DC weights

```
Input: $D^*, \Phi, S$                    ▷ True instance, DCs, schema sequence
Input: $\sigma_w, T_w, L_w$              ▷ Noise scale/#iteration/sample size in $\Psi$
Input: $b_w, S_w$                        ▷ Batch size in $\Psi$, sensitivity (Lemma 1)
1: procedure LEARNWEIGHT($D^*, \Phi, S, M, \Psi$)
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
```

with domain size of 18k. The training sample of size $b \times T$ in Algorithm 8 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.

4.3 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 11 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 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. (2.1). 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 $\mathcal{O}(n)$ 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 $\mathcal{O}(|\Phi|n^2)$ to $\mathcal{O}(|\Phi|L_w^2)$.

Lemma 1. *The L_2 sensitivity of the violation matrix for Φ that contains only unary and binary DCs is $S_w = |\phi_u| + |\phi_b| \times \sqrt{L_w^2 - L_w}$, where $|\phi_u|$ and $|\phi_b|$ represent the number of unary DCs and the number of binary DCs in Φ, respectively.*

Proof. Consider a pair of neighboring instances by changing one tuple. If a DC is an unary DC, then the differing tuple can change the violation count by 1. If the DC is a binary DC, the differing tuple may violate all other $L_w - 1$ tuples in the instance. Thus, the L_2 norm of the maximum violation count change S_w is:

$$\begin{aligned} S_w &= (|\phi_u| \times \sqrt{1}) + (|\phi_b| \times \sqrt{1^2 + 1^2 + \cdots + 1^2 + (L_w - 1)^2}) \\ &= |\phi_u| \times 1 + |\phi_b| \times \sqrt{L_w - 1 + (L_w - 1)^2} \\ &= |\phi_u| + |\phi_b| \times \sqrt{L_w^2 - L_w} \end{aligned}$$ □

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 § 4.4. The time complexity of this post-processing step is $\mathcal{O}(|\Phi|bT_w)$ in terms of the number of tuples that are used for learning.

4.4 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 8 Line 2-4);

M_2: Training $k-1$ discriminative models (Algorithm 8 Line 6-20);

M_3: Learning the DC weights if unknown (Algorithm 11).

Each process has been privatized using the Gaussian mechanism or DPSGD. The other steps (Algorithm 9 and Algorithm 10) not accessing the true database do not incur privacy loss. Hence, we can show KAMINO achieves DP by simple sequential composition [51] and post-processing property [52] of DP. However, this does not give us the tightest privacy bound. Instead, we apply *Rényi DP* (RDP) [128], a generalized privacy notion and its advanced composition techniques for the privacy analysis of KAMINO.

Definition 10 (Rényi-DP [128])**.** *A randomized algorithm $\mathcal{M}$ with domain $\mathcal{D}$ is (α,ϵ)-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 $\mathcal{M}(D)$ and $\mathcal{M}(D')$, respectively. It holds that: $\frac{1}{\alpha-1}\log \mathbb{E}_{x\sim\mathcal{M}(D')}\left(\frac{P_D(x)}{P_{D'}(x)}\right)^{\alpha} \leq \epsilon$.*

We state the RDP cost of KAMINO as follows. Both the post-processing and composability properties apply to RDP. Specifically, if a sequence of adaptive mechanisms $\mathcal{M}_1$, $\mathcal{M}_2$, $\cdots$, $\mathcal{M}_k$ satisfy (α,ϵ_1)-, (α,ϵ_2)-, $\cdots$, (α,ϵ_k)-RDP, then the composite privacy loss is $(\alpha, \sum_{i=1}^{k}\epsilon_i)$-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) [129].

Lemma 2. *Given a database D and query $f : \mathcal{D} \to \mathbb{R}^d$, returning $f(\{x \in D \mid x \text{ is sampled with probability } r\}) + \mathcal{N}(0, S_f^2\sigma^2\mathbb{I}^d)$ results in the following RDP cost for an integer moment α* [1]

$$R_{\sigma,r}(\alpha) = \begin{cases} \frac{\alpha}{2\sigma^2} & r = 1 \\ \sum_{k=0}^{\alpha}\binom{\alpha}{k}(1-r)^{\alpha-k}r^k\exp(\frac{\alpha^2-\alpha}{2\sigma^2}) & 0 < r < 1 \end{cases}$$

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

[1]Analysis on general fractional moments is in related work [129].

Theorem 9. *The total RDP cost of* Kamino *with parameter configuration set* $\Psi = \{\sigma_g, \sigma_d, \sigma_w, b, T, k, L_w, i_w \ldots\}$ *(Algorithm 7) is*

$$\begin{aligned} R_\Psi(\alpha) =& \frac{\alpha}{2\sigma_g^2} + T(k-1) \times \sum_{k=0}^{\alpha} \binom{\alpha}{k} (1-\frac{b}{n})^{\alpha-k} (\frac{b}{n})^k \exp(\frac{\alpha^2-\alpha}{2\sigma_d^2}) \\ &+ i_w \sum_{k=0}^{\alpha} \binom{\alpha}{k} (1-\frac{L_w}{n})^{\alpha-k} (\frac{L_w}{n})^k \exp(\frac{\alpha^2-\alpha}{2\sigma_w^2}), \end{aligned}$$

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

By the composition property [128] of RDP, the total RDP cost is:

$$\begin{aligned} R_{\text{Kamino}}(\alpha) =& \frac{\alpha}{2\sigma_g^2} + \sum_{k=0}^{\alpha} \binom{\alpha}{k} (1-\frac{L_w}{n})^{\alpha-k} (\frac{L_w}{n})^k \exp(\frac{\alpha^2-\alpha}{2\sigma_w^2}) + \\ & T(k-1) \times \sum_{k=0}^{\alpha} \binom{\alpha}{k} (1-\frac{b}{n})^{\alpha-k} (\frac{b}{n})^k \exp(\frac{\alpha^2-\alpha}{2\sigma_d^2}) \end{aligned}$$

□

By the tail bound property of RDP [128], we can convert the RDP cost of Kamino to (ϵ, δ)-DP, where ϵ is computed by

$$\epsilon_\Psi(\delta) = \min_\alpha R_\Psi(\alpha) + \frac{\log(1/\delta)}{\alpha-1}, \tag{4.5}$$

for a given δ. The order α is usually searched within a range [163].

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,

Algorithm 12 Searching DP parameters

```
Input: $\epsilon, \delta, \mathcal{D}, S$                    ▷ Privacy budget, domain, schema sequence
1: procedure SearchDParas($\epsilon, \delta, \mathcal{D}, S$)
2:     $C \leftarrow 1, \sigma_d \leftarrow 1.1, \eta \leftarrow 10^{-4}$                ▷ norm clip, noise scale, lr
3:     $\sigma_g \in [0.1/|\mathcal{D}(S[1])|, 4\sqrt{\log(1.25/\delta)}/\epsilon]$, $\sigma_d \in [1, 1.5]$
4:     $b \in [16, 32]$, $T \in [n/\min(b), 5n/\min(b)]$
5:     Initialize $\sigma_g, \sigma_d$ to the minimal, and $T, b$ to the maximal
6:     if DC weights unknown then
7:         $\epsilon_w, L_w \leftarrow 100$, $\sigma_w \leftarrow \sqrt{2\log(1.25/\delta_w)}/\epsilon_w$
8:         $b_w \leftarrow 1$, $T_w \leftarrow L_w/b_w$
9:     end if
10:    while $\epsilon_\Psi(\delta) > \epsilon$ do                               ▷ Eqn. (4.5)
11:        If $T > T_{min}$, then decrease T
12:        If $\sigma_d < \sigma_{dmax}$, then increase $\sigma_d$
13:        If $\sigma_g < \sigma_{gmax}$, then increase $\sigma_g$
14:        If $b > b_{min}$, then decrease $b$
15:    end while
16:    return $\Psi$, a set consisting of all above parameters
17: end procedure
```

Kamino provides a parameter search algorithm, summarized in Algorithm 12. 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 [27, 163] 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 5). If this privacy cost of this configuration is higher than ϵ (Line 10), then we use a priority order to decide which parameter to tune (Lines 11-14). This process is repeated till the privacy loss is capped at total budget. The time complexity is linear to the size of parameter space.

4.5 Evaluation

In this section, we evaluate the synthetic data generated by Kamino with three utility metrics: 1) consistency with DC constraints in the true data; 2) usefulness in training classification models; and 3) accuracy in answering α-way marginal queries. We show:

- Kamino preserves data consistency, while state-of-the-art methods fail to preserve most

Table 4.1: 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	$\approx 2^{52}$	Yes	$\phi_1^a : \neg(t_i[edu] = t_j[edu] \wedge t_i[edu\_num] \neq t_j[edu\_num])$ $\phi_2^a : \neg(t_i[cap\_gain] > t_j[cap\_gain] \wedge t_i[cap\_loss] < t_j[cap\_loss])$
BR2000	38,000	14	$\approx 2^{16}$	No	$\phi_1^b : \neg(t_i[a13] = t_j[a13] \wedge t_i[a11] < t_j[a11] \wedge t_i[a3] > t_j[a3])$ $\phi_2^b : \neg(t_i[a12] \neq t_j[a12] \wedge t_i[a13] \leq t_j[a13] \wedge t_i[a5] \geq t_j[a5])$ $\phi_3^b : \neg(t_i[a5] \leq t_j[a5] \wedge t_i[a3] > t_j[a3] \wedge t_i[a12] \neq t_j[a12] \wedge t_i[a11] > t_j[a11])$
Tax	30,000	12	$\approx 2^{71}$	Yes	$\phi_1^t : \neg(t_i[zip] = t_j[zip] \wedge t_i[city] \neq t_j[city])$ $\phi_2^t : \neg(t_i[areacode] = t_j[areacode] \wedge t_i[state] \neq t_j[state])$ $\phi_3^t : \neg(t_i[zip] = t_j[zip] \wedge t_i[state] \neq t_j[state])$ $\phi_4^t : \neg(t_i[state] = t_j[state] \wedge t_i[has\_child] = t_j[has\_child] \wedge t_i[child\_exemp] \neq t_j[child\_exemp])$ $\phi_5^t : \neg(t_i[state] = t_j[state] \wedge t_i[marital] = t_j[marital] \wedge t_i[single\_exemp] \neq t_j[single\_exemp])$ $\phi_6^t : \neg(t_i[state] = t_j[state] \wedge t_i[salary] > t_j[salary] \wedge t_i[rate] < t_j[rate])$
TPC-H	20,000	9	$\approx 2^{42}$	Yes	$\phi_1^h : \neg(t_i[c\_custkey] = t_j[c\_custkey] \wedge t_i[c\_nationkey) \neq t_j[c\_nationkey]$ $\phi_2^h : \neg(t_i[c\_custkey] = t_j[c\_custkey] \wedge t_i[c\_mktsegment) \neq t_j[c\_mktsegment]$ $\phi_3^h : \neg(t_i[c\_custkey] = t_j[c\_custkey] \wedge t_i[n\_name) \neq t_j[n\_name]$ $\phi_4^h : \neg(t_i[n\_name) = t_j[n\_name] \wedge t_i[n\_regionkey] \neq t_j[n\_regionkey])$

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.

4.5.1 Evaluation Setup

Datasets. We choose 4 different datasets with mixed data types and DCs, listed in Table 4.1. First, the Adult dataset [49] consists of 15 census attributes and 2 hard DCs. Second, the BR2000 dataset [174] has a smaller domain size than the Adult dataset, but it has 3 soft DCs with unknown weights. The third dataset, Tax [43], has a very large domain size, e.g., *zip* ($\approx 2^{15}$) and *city* ($\approx 2^{14}$) and 6 hard DCs. Last, TPC-H [9], 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 [174], a statistical method based on Bayesian network; 2) PATE-GAN [95], a GAN-based method that trains a data generator using the PATE's student-teacher model [140]; 3) DP-VAE [38], which samples from the latent space of a privately trained auto-encoder [100]; and 4) The winning solution of the NIST challenge [133] (labeled as NIST), which applies probabilistic inference [125] 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 [61]. 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, $h : \mathcal{D} \rightarrow \mathbb{R}^{|\mathcal{D}(\mathbb{A})|}$ on the synthetic data D' and true data D^*, respectively, and then report the total variation distance [158] 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 from AimNet in the HoloClean[2] system. For the baselines (PrivBayes[3], PATE-GAN[4], DP-VAE and NIST[5]), we reused the code from their authors with all default parameters. All the 9 models in the learning task were implemented using standard libraries [8, 40] and trained

[2]https://github.com/HoloClean/holoclean/

[3]https://sourceforge.net/projects/privbayes/

[4]https://bitbucket.org/mvdschaar/mlforhealthlabpub/src/master/alg/pategan/

[5]https://github.com/usnistgov/PrivacyEngCollabSpace/tree/master/tools/de-identification/Differential-Privacy-Synthetic-Data-Challenge-Algorithms/rmckenna

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.

4.5.2 End-to-End Evaluation

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

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 4.2 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.

Experiment 2: Model Training

Figure 4.3 shows the accuracy and F1 on classifying all attributes. Each data point in Figure 4.3 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 4.3 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).

Table 4.2: 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**

Experiment 3: α-way Marginals

Figure 4.4 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).

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,

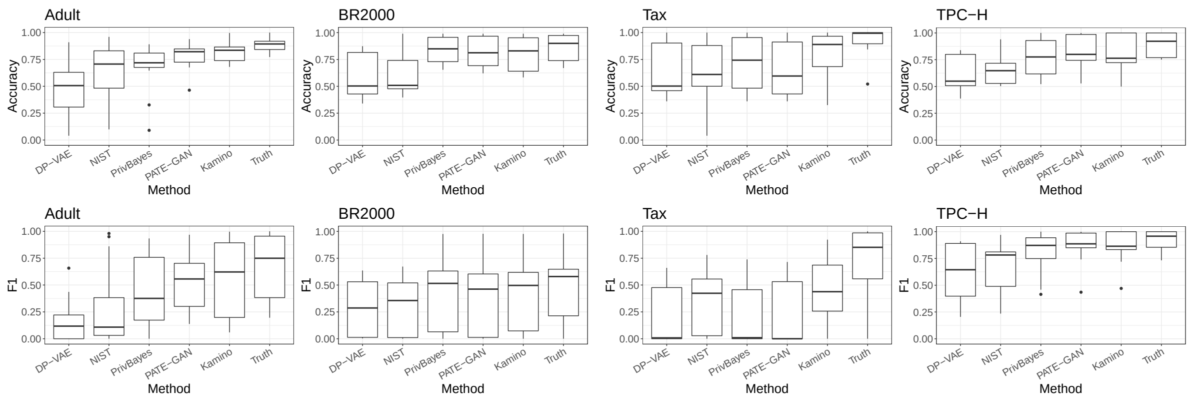


Figure 4.3: Accuracy and F1 of evaluating classification models, which are tested on the true dataset and trained on synthetic data by different methods. Each point represents an averaged classification quality (accuracy or F1) over 9 models for one target attribute using all other attributes as features. 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.

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 4.7 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 4.7 shows, performance of KAMINO is dominated by training and sampling, which together take more than 99% of the total time.

4.5.3 Component Evaluation

Experiment 5: Effectiveness of constraint-aware components

Recall that our approach takes DCs into account when it samples synthetic values (§ 4.2.2) and generate the schema sequence (§ 4.2.3). In this experiment, we compare KAMINO with three sub-optimal KAMINO that do not have the constraint-aware components:

- Replace constraint-aware sampling (Algorithm 9) in KAMINO with sampling tuples independently, labeled as "RandSampling";
- Replace constraint-aware sequencing (Algorithm 10) by a random sequence, labeled as "RandSequence";

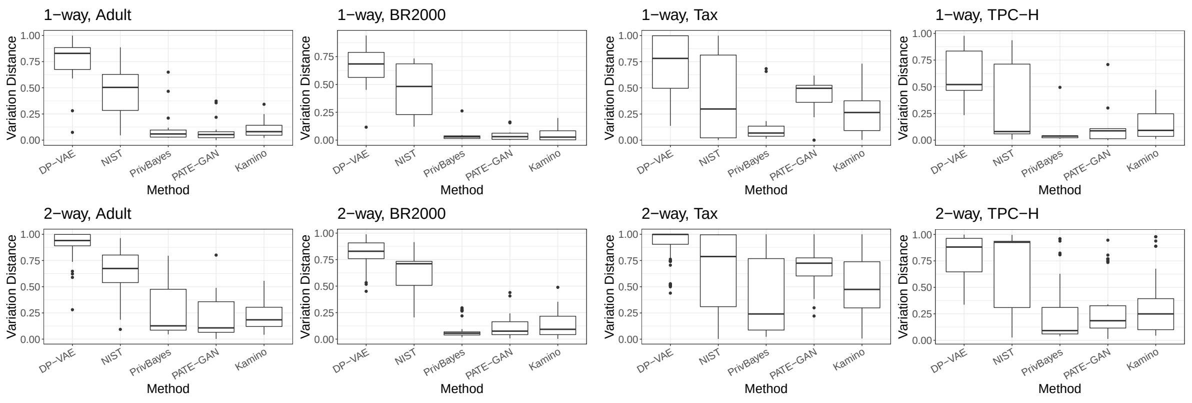


Figure 4.4: Total variation distance on α-way marginals, where $\alpha = [1, 2]$. Each point represents a total variance distance for one attribute set, and each box represents total variance distance for all attribute sets. It shows that KAMINO can achieve overall the best (Adult) or close to the best (BR2000, Tax and TPC-H) variation distance.

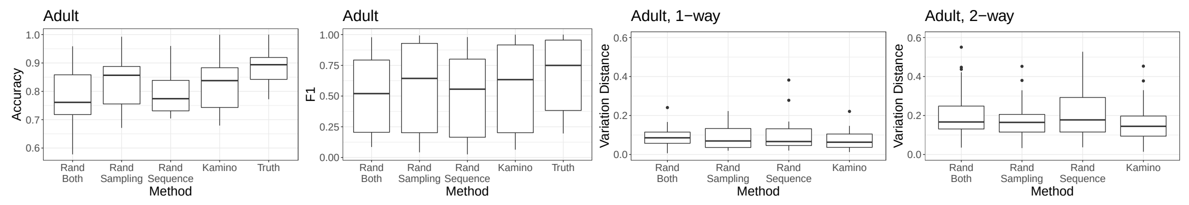
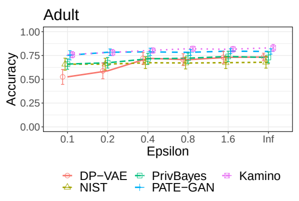

(a) Accuracy (b) F1 (c) 1-way marginal (d) 2-way marginal

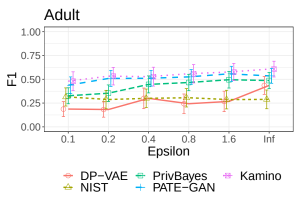
Figure 4.5: Accuracy and F1 of model training on KAMINO, and sub-optimal KAMINO without constraint-aware sampling, sequencing, and neither, using the Adult dataset as the example. It shows the the KAMINO with constraint-aware components can achieve the best quality in both the learning task and in the query task.

- Replace both components above, labeled as "RandBoth".

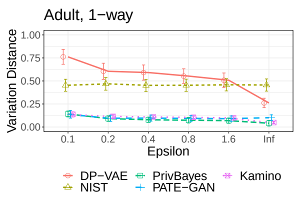
Table 4.3 compares DC violations of the synthetic data generated by KAMINO and by sub-optimal KAMINO without constraint-aware components. First, we see that without constraint-aware sampling component (Algorithm 9), the synthetic data generated by RandSampling and RandBoth have more violations than the other two methods. Second, the constraint-aware sequencing component (Algorithm 10) 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



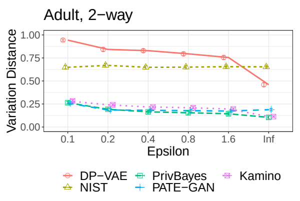
(a) Accuracy



(b) F1



(c) 1-way marginal



(d) 2-way marginal

Figure 4.6: Task quality of the KAMINO and baselines by varying privacy budget $(\epsilon, 10^{-6})$.

Table 4.3: 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

constraint-aware components, quality downgrades in both learning and query task shown in Figure 4.5.

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.

Experiment 6: Kamino vs Accept-Reject Sampling

KAMINO's constraint-aware sampling (Algorithm 9) 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 [134], 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

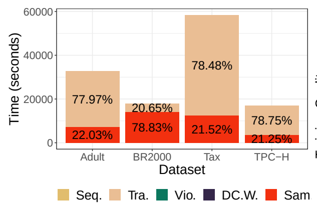
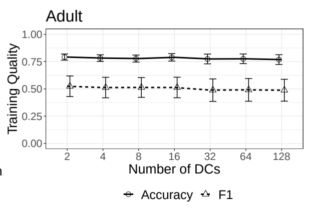
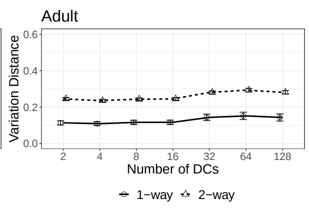


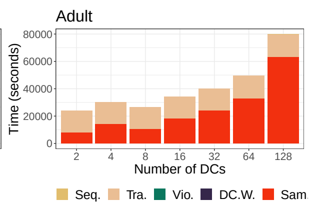
Figure 4.7: Time of end-to-end runs on all datasets.



(a) Model Training



(b) Marginal Distance



(c) Time Complexity

Figure 4.8: Varying the number of 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.

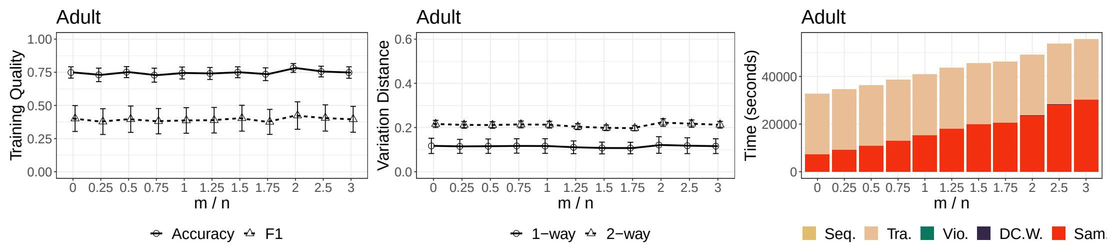
On the BR2000 dataset with soft DCs, KAMINO with AR-sampling completes in 1.26 hours (0.24 hour for the AR-sampling step) on average, which is faster than the constraint-aware sampling (3.9 hours). AR-sampling converges faster due to its relatively high accept ratio. For DC violations and task qualities,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

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 4.6 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 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 4.3-4.4, KAMINO always achieves the best in training quality (Figures 4.6a-4.6b) and close to best marginal distances (Figures 4.6c-4.6d) 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 $\epsilon = 1$.

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 [142] to simulate the knowledge from the domain expert.



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

Figure 4.9: Task quality and execution time by varying the number of resampling per attribute.

Figure 4.8 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 11), increasing the number of DCs implies more noisy adjustment for the sampling probabilities (Algorithm 9). As a result, task quality is expected to decrease given a finite privacy budget. Figure 4.8a and Figure 4.8b 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 9), 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 4.8c shows that when the number of DC increases from 2 to 128, the total execution time increases only by 3×.

Experiment 9: Varying the Number of Re-sampling in MCMC

Recall that Kamino's sampling process adopts MCMC by re-sampling m random cells after each column is synthesized (Algorithm 9). Figure 4.9 shows the effects of m, which is represented as a ratio over dataset cardinality n on the x-axis. Comparing to no re-sampling, the re-sampling up to $m = 3n$ using the same probabilistic data model improves accuracy (by up to 0.03), F1 (by up to 0.02), and both 1-way and 2-way marginal distances (by 0.01 and 0.02, respectively). Meanwhile, more re-sampling requires longer execution time (by up to 4×).

Experiment 10: Efficiency Optimizations

This section presents two optimization techniques to speed up Kamino under certain conditions. The first technique is to train Kamino's probabilistic data models (Algorithm 8) in parallel, where each model $M_{X,y}$ is trained on a separate machine and tuple embeddings are initialized randomly, instead of reusing previously trained ones. Without reusing the tuple embeddings, we observe that although all task qualities drop 0.01 on the Adult dataset, the training time becomes $3.5\times$ faster.

The second optimization is to exploit the special property of DCs. One special DC is the hard functional dependencies (e.g., keys), where the right-hand-side attribute has only one unique value given the left-hand-side values. Instead of checking violations for the set of candidate values, we can find the correct value from previously synthesized data. We scale up the TPC-H dataset to 1 million rows with the same set of DCs. Kamino can complete in 10 hours by leveraging the fact that all DCs are hard functional dependencies.

4.6 Discussion and Conclusion

4.6.1 DC Violation Analysis

Consider in the non-private setting and assume that the true database has no violations for the given set of DCs. Intuitively, we can learn an accurate probabilistic database model according to the learnability theorem (Theorem 14 [151]). When sampling from the learned probabilistic database model, the sampled synthetic database instance should have a small number of DC violations. Formally, we state the following theorem.

Theorem 10. *Given a true database instance D^* with no constraint violations for a set of DCs Φ, we learn the probabilistic database model $\mathbb{D}$ and sample a synthetic database instance $D' \sim \mathbb{D}$ based on Eqn. (4.2). The sampled D' incurring any DC violations has a low probability.*

Proof. Let Θ represent the parameter set for the tuple probability, and W denote the weight vector of DCs. Given the true database instance D^* with no DC violations, the parameters can be found by maximizing the likelihood of D^*, i.e.,

$$\Theta^*, W^* = \arg\max_{\Theta, W} \frac{\Pi_{t \in D^*} \Pr(t; \Theta)}{Z} \tag{4.6}$$

where

$$Z = \sum_{D \in \mathbf{D}_{V=0}} \Pi_{t \in D} \Pr(t; \Theta) + \sum_{D \in \mathbf{D}_{V \neq 0}} \Pi_{t \in D} \Pr(t; \Theta) \exp(-\sum_{\phi \in \Phi} w_\phi |V(\phi, D)|)$$

and $\mathbf{D}_{V=0}$ represents the set of database instances with no DC violations. To maximize the likelihood of D^*, the weight w_ϕ for each DC are set to be very large (i.e., $w_\phi \to \infty$).

Finally, suppose $\exists \phi \in \Phi$, such that $|V(\phi, D')| \neq 0$, and consider the probability of sampling D':

$$\Pr(D'; \Theta^*, W^*) = \frac{\Pi_{t \in D'} \Pr(t) \times \exp(-\sum_{\phi \in \Phi} w_\phi |V(\phi, D')|)}{Z} \tag{4.7}$$

Since $w_\phi \to \infty$ (Eqn. (4.6)) and $|V(\phi, D')| \neq 0$, then $\Pr(D'; \Theta^*, W^*) \to 0$. $\square$

According to Theorem 10, the synthetic database instance D' is most likely to have no violations. In § 4.5.2, we empirically show that $|V(\phi, D')| = 0$ for all hard DCs on all output instances.

More generally, even if the true database has a small number of DC violations (i.e., the low-noise condition [151]), we are still able to learn the parameters of the probabilistic database model $\mathbb{D}$ accurately. In addition, the sampling process follows the chain rule (Eqn. (4.4)) for Eqn. (4.2) and hence, allows an instance to be sampled correctly from $\mathbb{D}$. As a result, the true and the synthetic database instances come from the same distribution. The probability of a synthetic instance depends on the the softness of the DCs (Eqn. (4.7)).

Finding the error bound of DC violations for general DCs with differential privacy guarantee is an exciting future work.

4.6.2 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.

Chapter 5

SMFD: Secure Multi-Party Functional Dependency Discovery

5.1 Problem Statement and Solution Overview

5.1.1 Problem Statement

Assume a dataset D in schema R, which is horizontally partitioned into D_1 to D_m: $D = \cup_{i=1}^{m} D_i$. Let S denote the set of all FDs that are valid on the dataset D: $S = \{f \mid D \models f\}$, and let S_i be the set of all FDs that are valid on a partition D_i: $S_i = \{f \mid D_i \models f\}$. Let $\hat{S}$ represent the intersection of all S_i: $\hat{S} = \cap_{i=1}^{m} S_i$. It is clear that $S \subseteq \hat{S}$, as we showed in the example from § 1.

Recall in § 2.1.1, an FD $A \rightarrow B$ is *trivial* when $B \in A$. We also call an FD *minimal* if $\nexists K \subseteq A$ such that $A \setminus K \rightarrow B$. Let $\sum$ be the set of all non-trivial and minimal FDs on D, $\sum \models S$, i.e., any $f \in S$ can be either in $\sum$ or implied by $\sum$.

The problem is to securely find $\sum$ against semi-honest adversaries, where each honest-but-curious party $\mathbb{P}_i (i \in [1, m])$ owns a private D_i and follows the protocol honestly, but tries to infer information from other parties.

5.1.2 A Naïve Attempt

One might think of solving the problem by intersecting FDs from different partitions. We use the following case as a counter example.

Example 9: Consider the following FDs:

$f_1 : edu \rightarrow edu\_num$ $\qquad f_2 : country \rightarrow income$

f_1 states that for any two persons with the same edu, they must have the same edu_num. f_2 states $country$ determines $income$. Because f_1 and f_2 are valid FDs on individual partition D_1 and D_2, $f_1, f_2 \in S_1$ and $f_1, f_2 \in S_2$, and hence $f_1, f_2 \in \hat{S} = S_1 \cap S_2$. By examining the tuples from individual partitions, it is clear that f_1 and f_2 are valid FDs on both D_1 and D_2.

D_1

	edu	edu_num	marital	occupation	country	income
t_1	Bachelors	13	Single	Exec-mag	Canada	>50K
t_2	Bachelors	13	Married	Exec-mag	Canada	>50K
t_3	HS-grad	9	Divorced	Exec-mag	Canada	>50K
t_4	11th	7	Married	Handlers	USA	≤50K
t_5	Bachelors	13	Married	Sales	USA	≤50K

D_2

	edu	edu_num	marital	occupation	country	income
t_1	HS-grad	9	Divorced	Sales	Canada	≤50K
t_2	HS-grad	9	Single	Craft-repair	Canada	≤50K
t_3	As-acdm	12	Single	Server	Canada	≤50K
t_4	Bachelors	13	Divorced	Exec-mag	Canada	≤50K
t_5	11th	7	Married	Exec-mag	USA	>50K

Figure 5.1: Two partitions of employee dataset.

However, by examining the tuples from $D = D_1 \cup D_2$, f_2 is not valid on D, due to violations such as $D_1.t_1[country] = D_2.t_1[country]$, but $D_1.t_1[income] \neq D_2.t_1[income]$. □

5.1.3 Solution Overview

Figure 5.2 depicts the architecture of our solution. Our framework adopts a top-down approach [120] to prune the FD search space. Assume there are m parties numbered from

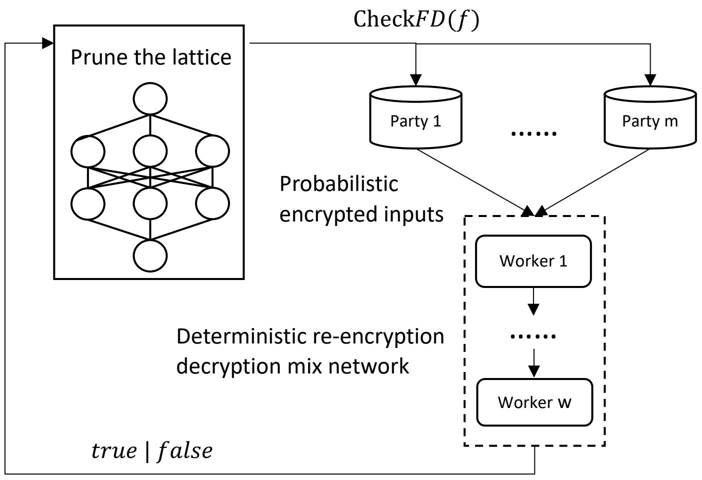


Figure 5.2: Architecture of secure multi-party FD discovery.

1 to m, each holding a database partition $D_i (i \in [1, m])$, and a mix network chain consisting of w workers sequentially from worker 1 to w.

We first construct a set containment lattice. Each node in the lattice represents a set of attributes, and every edge in the lattice represents a candidate FD [86]. In order to prune the space of possible FDs, for each candidate FD f, all the m parties jointly validate the candidate FD, which is represented by the functionality of $CheckFD(f)$. The validation process involves two types of actions: 1) all the parties prepare necessary inputs and encrypt using probabilistic encryption; and 2) the encrypted messages are sent to a deterministic re-encryption decryption mix network. A mix network which consists of a chain of w workers, securely computes on the inputs and and eventually returns the validity of f, which will be used to prune the lattice in the next round. The process runs until all the edges have been properly traversed.

Note that traversing the lattice can be guided by any schema driven search algorithms [12, 86, 132, 170], and is not the contribution of this work. The scope of this work focuses on the fundamental functionality $CheckFD(f)$ that is used to prune the candidate space.

The details of the solution will be discussed as follows: § 5.2 explains the $CheckFD(f)$ flow and the end-to-end solution. Two deterministic re-encryption decryption mix networks, which are used together to securely validate candidate FDs will be introduced in § 5.3.

5.2 Secure FD Discovery

Recall in § 5.1.3 that our solution is based on validating candidate FD to prune the search space. We first formulate the distribuetd FD validation in § 5.2.1, and then introduce the FD discovery protocol in § 5.2.2.

5.2.1 Distributed FD Validation

The main observation is that in order to validate a candidate FD f, one needs to compare all the attributes relevant to f across the partitions. Based on traditional FD validation on a single dataset [86], we can extend the FD validation rule on distributed partitions as follows:

Lemma 3. *(Distributed FD Validation) A dataset D in schema R is partitioned into D_1 to D_m: $D = \cup_{i=1}^{m} D_i$. An FD $A \rightarrow B$ holds on D if and only if $e(A^D) = e((A \cup B)^D)$.*

Example 10: Consider $D = D_1 \cup D_2$ in Figure 5.1. For the candidate FD f_1: $\pi_{edu} = \{$`Bachelors, HS-grad, 11th, As-acdm`$\}$, and hence $e(edu^D) = 10 - 4 = 6$. Similarly, $\pi_{edu,edu\_num} = \{$`(Bachelors,13),(HS-grad,9),(11th,7),(As-acdm,12)`$\}$. and $e((edu, edu\_num)^D) = 6$. Therefore, $D \models f_1$.

However, for the candidate FD f_2: $e(ethnicity^D) \neq e((ethnicity, income)^D)$; therefore, $D \not\models f_2$. □

Based on Lemma 3, we can further formalize the attribute partition error over multiple parties. For any attribute set $A \subseteq R$, the cardinality of m-set union[1] satisfies:

$$\|\pi_A^D\| = \sum_{i=1}^{m} \|\pi_A^{D_i}\| \tag{5.1}$$

Meanwhile, the size of attribute partitions satisfies:

$$\begin{aligned} |\pi_A^D| = & \sum_{i=1}^{m} |\pi_A^{D_i}| - \sum_{1 \leq i < j \leq m} |V_A^{D_i} \cap V_A^{D_j}| \\ & + \sum_{1 \leq i < j < k \leq m} |V_A^{D_i} \cap V_A^{D_j} \cap V_A^{D_k}| + \cdots \\ & + (-1)^{m-1} |\bigcap_{i=1}^{m} V_A^{D_i}| \end{aligned} \tag{5.2}$$

[1]We assume each tuple has its own tuple id and partition id; therefore no identical tuples exist.

Where $V_A^{D_i}$ is the set of values of attribute partition of attribute set A on partition D_i. Eqn. (5.1) − Eqn. (5.2) leads to:

$$\begin{aligned} e(A^D) = &\sum_{i=1}^{m} e(A^{D_i}) + \sum_{1 \le i < j \le m} |V_A^{D_i} \cap V_A^{D_j}| \\ &- \sum_{1 \le i < j < k \le m} |V_A^{D_i} \cap V_A^{D_j} \cap V_A^{D_k}| + \cdots \\ &+ (-1)^m |\bigcap_{i=1}^{m} V_A^{D_i}| \end{aligned} \tag{5.3}$$

From Eqn. (5.3), the final attribute error consists of two parts: 1) the sum of attribute errors on each partition $\sum_{i=1}^{m} e(A^{D_i})$; and 2) the rest of power set intersection cardinality (PSI-CA). Every party $\mathbb{P}_i$ is able to compute $e(A^{D_i})$ and $V_A^{D_i}$ locally since D_i belongs to $\mathbb{P}_i$. To validate an FD $A \to B$, one simply needs to compute and compare the equality of $e(A^D)$ and $e((A \cup B)^D)$ based on Eqn. (5.3).

5.2.2 A Secure FD Discovery Protocol

We arbitrarily choose one party to act as the *moderator*, who drives a classical schema-driven FD discovery algorithm such as TANE [86], and instructs FD validations.

Figure 5.3 illustrates the protocol to securely fulfil the $CheckFD(f)$ function: given a candidate FD $f : A \to B$ as input, the protocol consists of three steps: 1) on attribute sets A and AB, it computes the power set intersection cardinality (PSI-CA) if not done yet. Each party encrypts and sends its set of values for the attribute partition via a multiplicative deterministic re-encryption decryption mixnet (§ 5.3.1). The output of the mixnet is the encrypted set of values for the attribute partition, based on which, the moderator intersects all sets directly over encrypted values for both attribute sets A and AB, and returns the cardinalities; 2) each party computes local attribute partition errors on A and AB and encrypts them to send the difference to an additive deterministic re-encryption decryption mixnet (§ 5.3.2). Meanwhile, the moderator also encrypts and sends the difference of PSI-CA of attribute sets A and AB as input. The output of the mixnet is the encrypted error difference; finally 3), the moderator concludes the validity of candidate FD based on the output of second mixnet: $A \to B$ is true if and only if the encrypted value equals to 1.

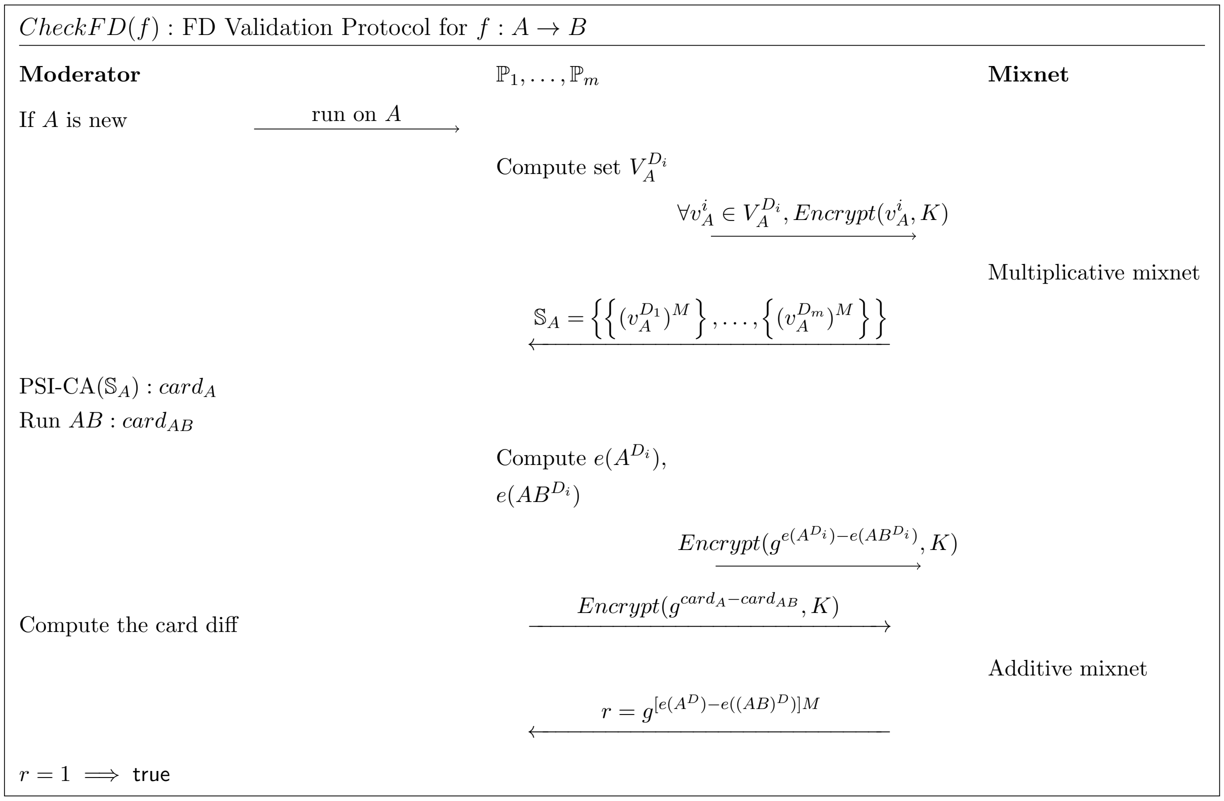

$CheckFD(f)$: FD Validation Protocol for $f : A \to B$
Moderator
$\mathbb{P}_1, \ldots, \mathbb{P}_m$
Mixnet
If A is new
run on A
Compute set $V_A^{D_i}$
$\forall v_A^i \in V_A^{D_i}, Encrypt(v_A^i, K)$
Multiplicative mixnet
$\mathbb{S}_A = \left\{\left\{(v_A^{D_1})^M\right\}, \ldots, \left\{(v_A^{D_m})^M\right\}\right\}$
PSI-CA($\mathbb{S}_A$) : $card_A$
Run AB : $card_{AB}$
Compute $e(A^{D_i})$, $e(AB^{D_i})$
$Encrypt(g^{e(A^{D_i})-e(AB^{D_i})}, K)$
Compute the card diff
$Encrypt(g^{card_A - card_{AB}}, K)$
Additive mixnet
$r = g^{[e(A^D)-e((AB)^D)]M}$
$r = 1 \implies$ true

Figure 5.3: The FD validation protocol for securely validating a candidate FD using mixnets.

The two mixnets in above protocol will be explained in details later in § 5.3.1 and § 5.3.2, respectively. In the rest of this section, we present an efficient approach to compute PSI-CA.

A naïve way to compute PSI-CA is that for each set intersection, we intersect the encrypted values directly from the sets. Given there are m partitions, the number of intersections is $2^m - m - 1$, which is expensive.

To reduce the exponential cost of PSI-CA, Algorithm 13 presents a linear approach to efficiently compute PSI-CA for a set of m sets. The intuition is that rather than computing intersections, we count the contribution of each value to PSI-CA, which is determined by the number of its occurrence in all partitions. For example, if an value occurs three times in any of the three partitions, then that value will contribute to PSI-CA by being intersected in

Algorithm 13 Power set intersection cardinality

```
Input: $\mathbb{S}$                                   ▷ set of $m$ sets to be intersected
 1: procedure PSI-CA($\mathbb{S}$)
 2:     $card \leftarrow 0$
 3:     $imap < int, int >$                           ▷ (key, occurrance)
 4:     for $t \in [1, m], \forall k \in \mathbb{S}[t]$ do
 5:         $imap[k] + +$                             ▷ increase the counter
 6:     end for
 7:     $cmap < int, int >$                           ▷ (occurrance, contribution)
 8:     for all $occ \in imap.values$ do
 9:         if $occ \geq 2$ then                      ▷ element occurs than once
10:             if $\neg cmap.contains(occ)$ then
11:                 $cntrbtn \leftarrow 0$
12:                 for $i \leftarrow occ; i \geq 2; --i$ do
13:                     $c \leftarrow C^{i}_{occ}$            ▷ $i$-combination
14:                     if $i$ mod 2 == 0 then        ▷ flipping signs
15:                         $cntrbtn \leftarrow cont + c$
16:                     else
17:                         $cntrbtn \leftarrow cont - c$
18:                     end if
19:                 end for
20:                 $cmap(occ) \leftarrow cntrbtn$
21:             end if
22:             $card \leftarrow card + cmap.get(occ)$
23:         end if
24:     end for
25:     return $card$
26: end procedure
```

one 3-set intersection and three 2-set intersections. This method simply requires traversing all the elements only once. We use two simple map structures to keep counting: 1) *imap* stores the mapping of encrypted elements as the keys, and values are the counter of key occurrences; 2) *cmap* stores the occurrence of an encrypted element as the key, and its contribution to the set intersection cardinalities in Eqn. (5.3) (§ 5.2.1).

Figure 5.4 gives an example to compute PSI-CA using Algorithm 13 with an input set $\mathbb{S}$ consisting of three sets, each of which is a set of encrypted values from a partition. In

$$\mathbb{S} = \{\{a,b,c\},\{a,b,d\},\{b,d\}\}$$

imap

element	occurrence
a	2
b	3
c	1
d	2

cmap

occurrence	contribution
2	$C_2^2 = 1$
3	$-C_3^3 + C_3^2 = 2$

$$card = 1 + 2 + 1 = 4$$

Figure 5.4: An example of computing PSI-CA.

§ 5.3, we will explain in details how each value is probabilistically encrypted but still can be deterministically compared for equality. Given the example input of $\mathbb{S}$ in Figure 5.4, we can easily reason out that the PSI-CA equals to $card = 4$ by Eqn. (5.3) (§ 5.2.1). With Algorithm 13, the process starts with a single scan on each elements in order to build the *imap* structure (Line 3—6). After the traverse, *imap* stores the occurrences for all values from partitions. For example, element a appears twice in $\mathbb{S}$. Then, *imap* is sequentially visited to build the *cmap* (Line 7—24), which stores the contribution of each occurrence to the final PSI-CA. For instance, the occurrence of element b is 3 in S, then it contributes to the overall cardinalities by showing up in one 3-set intersection $(-C_3^3)$, and three 2-set intersections (C_3^2). The *cmap* is adaptively built and can be reused. For instance, $(2, 1)$ was inserted when it visited $(a, 2)$ in *imap*, and can be reused when visiting $(d, 2)$. In the meantime of building *cmap*, the final cardinality is iteratively calculated (Line 22): $card = 1 + 2 + 1$.

In fact, the *cmap* can be built offline as a pre-processing step. Recall in Figure 5.2 (§ 5.1), the $CheckFD(f)$ is iteratively invoked. Although the elements (in encrypted format) of each run are different, the cardinality only depends on the occurrences regardless of the elements. Hence, computing PSI-CA only requires one-pass scan of all encrypted elements to construct the *imap*, while the *cmap* serves as a lookup table. Therefore, the complexity of PSI-CA is linear to the number of elements.

5.3 Equality-Aware Mixnet

In this section, we propose two versions of mixnets that enable secure equality testing on the input values. The first mixnet (§ 5.3.1) enables value-level equality check, which allows comparing the equality of input messages without revealing the messages themselves. The second mixnet (§ 5.3.2) supports comparing the sum of integer set to a given value. We will also prove the security of both mixnets.

5.3.1 Value-Level Equality Testing

The first mixnet is designed to achieve the following goal: given a set of input values $\{v_1, \ldots, v_m\}$, where v_i is the confidential message from party $\mathbb{P}_i$, and a boolean functionality $f_v(\cdot)$ taking two input values and outputting true if inputs are equal, the mixnet can securely fulfil $f_v(\cdot)$ on all pairs from the input set.

We now introduce the *multiplicative deterministic re-encryption decryption mixnet* using examples. Figure 5.5 shows one such mixnet with 3 workers labelled from $\mathbb{W}_1$ to $\mathbb{W}_3$ sequentially. There are 2 parties numbered as $\mathbb{P}_1$ and $\mathbb{P}_2$, with input values a' and b' respectively, which are to communicate via the mixnet. The input values are usually hashed into $\mathbb{G}_p$ using a hash function $H(\cdot)$. Initially (step 0.1), each worker generates a pair of keys (described in § 2.1.4) and distributes the public keys to all parties and workers. Then (step 0.2), each party fully encrypts its values using all the public keys before sending the encrypted values to the first worker of the mixnet (step 0.3).

Algorithm 14 Fully encrypting a single value

```
Input: $v$                                        ▷ an integer to be encrypted
Input: $K$                                         ▷ the set of all public keys
1: procedure ENCRYPT($v, K$)
2:     $(c_1, c_2) \leftarrow \mathsf{Enc}(c_2, \prod_{k \in K} k)$
3:     return $(c_1, c_2)$
4: end procedure
```

Algorithm 14 describes the encryption process that occurs on each party. All operations are performed in $\mathbb{G}_p$, and for simplicity we omit this in Algorithm 14 and in the rest of the description. The encrypted value is represented as a pair in the form of (c_1, c_2), which is encrypted under the product of the public keys. Intuitively, the encryption process is to encrypt multiply layers where each layer is secured by a key from one of the mixnet worker.

	Party $\mathbb{P}_1$	Party $\mathbb{P}_2$
0.1, Sync $K\{g^x, g^y, g^z\}$	$value : a = H(a')$	$value : b = H(b')$
0.2, Encrypt$(value, K)$	$(g^{r_1}, ag^{r_1x}g^{r_1y}g^{r_1z})$	$(g^{r_2}, bg^{r_2x}g^{r_2y}g^{r_2z})$
0.3, Send values to mixnet		$\Downarrow$
.... Worker $\mathbb{W}_1$. Keys: (x, g^x)		
1.0, Initialize	secret re-encryption key: $x' \leftarrow_{\$} \mathbb{Z}_p$	
1.1, Decrypt	$(g^{r_1}, ag^{r_1y}g^{r_1z})$	$(g^{r_2}, bg^{r_2y}g^{r_2z})$
1.2, Re-encrypt	$(g^{r_1x'}, a^{x'}g^{r_1x'(y+z)})$	$(g^{r_2x'}, b^{x'}g^{r_2x'(y+z)})$
1.3, Re-randomize	$s_1^1 \leftarrow_{\$} \mathbb{Z}_p, (g^{r_1x'+s_1^1}, a^{x'}g^{(r_1x'+s_1^1)(y+z)})$	$s_1^2 \leftarrow_{\$} \mathbb{Z}_p, (g^{r_2x'+s_1^2}, b^{x'}g^{(r_2x'+s_1^2)(y+z)})$
1.4, Permute and send to next		$\Downarrow$
.... Worker $\mathbb{W}_2$. Keys: (y, g^y)		
2.0, Initialize	secret re-encryption key: $y' \leftarrow_{\$} \mathbb{Z}_p$	
2.1, Decrypt	let $m_1^1 = r_1x' + s_1^1, (g^{m_1^1}, a^{x'}g^{m_1^1z})$	let $m_1^2 = r_2x' + s_1^2, (g^{m_1^2}, b^{x'}g^{m_1^2z})$
2.2, Re-encrypt	$(g^{m_1^1y'}, a^{x'y'}g^{m_1^1y'z})$	$(g^{m_1^2y'}, b^{x'y'g^{m_1^2y'z}})$
2.3, Re-randomize	$s_2^1 \leftarrow_{\$} \mathbb{Z}_p, (g^{m_1^1y'+s_2^1}, a^{x'y'}g^{(m_1^1y'+s_2^1)z})$	$s_2^2 \leftarrow_{\$} \mathbb{Z}_p, (g^{m_1^2y'+s_2^2}, b^{x'y'}g^{(m_1^2y'+s_2^2)z})$
2.4, Permute and send to next		$\Downarrow$
.... Worker $\mathbb{W}_3$. Keys: (z, g^z)		
3.0, Initialize	secret re-encryption key: $z' \leftarrow_{\$} \mathbb{Z}_p$	
3.1, Decrypt	let $m_2^1 = m_1^1y' + s_2^1, (g^{m_2^1}, a^{x'y'})$	let $m_2^2 = m_1^2y' + s_2^2, (g^{m_2^2}, b^{x'y'})$
3.2, Re-encrypt	$a^{x'y'z'}$	$b^{x'y'z'}$

Figure 5.5: A multiplicative, deterministic re-encryption decryption mixnet with 3 workers and 2 parties.

Each worker in the mixnet does the following operations:

0) Initialization (step .0). For each run, every worker generates a fresh secret re-encryption key that will be used for re-encryption in step .2.

1) Decryption (step .1). Since the value was encrypted with the worker's public key, the worker decrypts it using its private key. Consider the encrypted value from $\mathbb{P}_1$ in Figure 5.5, $\mathbb{W}_1$ uses its private key x and the random variable g^{r_1} and constructs a new random variable g^{r_1x}, which is then used to divide the encrypted value $ag^{r_1x}g^{r_1y}g^{r_1z}$ as a denominator. Hence, the encrypted value is decrypted to $ag^{r_1y}g^{r_1z}$.

2) Re-encryption (step .2). The decrypted value is deterministically re-encrypted by the secret re-encryption key. Continue above example, $\mathbb{W}_1$ re-encrypts $ag^{r_1 y}g^{r_1 z}$ by exponentiating with its secret re-encryption key x', to $a^{x'}g^{r_1 x' y}g^{r_1 x' z}$. Similarly, the random value is also updated.

3) Re-randomize the encrypted value (step .3). This is simply realized by freshly encrypting a value of 1 and homomorphically multiplying it to the cipher text.

4) Permute the values, and send to the next worker (step .4). This step adds an additional protection to hide the ordering. Each worker does the same set of operations until the values flow out from the last worker.

Algorithm 15 Mixneting a single value

```
Input: $(c_1, c_2)$                              ▷ cipher text to be re-encrypted
Input: $x$                                       ▷ private key of the running party
Input: $x'$                                      ▷ secret re-encryption key of the running party
1: procedure RE-ENCRYPT($(c_1, c_2)$)
2:     $c_2 = \mathsf{Dec}((c1, c2), x)$                       ▷ decrypt
3:     $(c_1, c_2) = (c_1^{x'}, c_2^{x'})$                     ▷ encrypt with re-encryption key
4:     $(c_1', c_2') = \mathsf{Enc}(1, \Pi)$                   ▷ $\Pi$: product of next pubkeys
5:     $(c_1, c_2) = (c_1 \times c_1', c_2 \times c_2')$       ▷ re-randomize
6:     return $(c_1, c_2)$
7: end procedure
```

Algorithm 15 describes the mixnet operations that occur on the workers. As the output of the mixnet, the input values are permuted and deterministically encrypted by the secret re-encryption keys of all workers. Thus, equality checking is realized by directly comparing the encrypted values. □

Let Π_v denote a multiplicative deterministic re-encryption decryption mixnet, we have the following theorem:

Theorem 11. *Π_v securely implements the functionality $f_v(\cdot)$ in the presence of semi-honest adversary given at least one party and one worker are not controlled by the adversary.*

Proof. The functionality $f_v(\cdot)$ is fulfilled by comparing the equality of encrypted values, and outputs true or false as the final output of the protocol. Correctness is immediate, so we proceed to the privacy proof. We consider that both the parties and the workers can be controlled by the semi-honest adversary. Let Sim be a simulator, $\mathcal{A}$ be a semi-honest adversary, and the input information regarding to the group be $I = \{\mathbb{G}_p, H(\cdot)\}$.

Suppose adversary $\mathcal{A}$ controls party $\mathbb{P}_1$, workers $\mathbb{W}_1$ and $\mathbb{W}_2$ using Figure 5.5 as the running example. The actual view of $\mathcal{A}$ consists of four messages m_{1-4}. In Figure 5.5, m_1 and m_2 correspond to the encrypted value $(g^{r_2}, bg^{r_2x}g^{r_2y}g^{r_2z})$ from honest $\mathbb{P}_2$, and m_3 and m_4 represent a value from the permuted, re-encrypted set $\{a^{x'y'z'}, b^{x'y'z'}\}$ from honest worker $\mathbb{W}_3$ (step 3.2), respectively. The task of Sim is to output four messages that are indistinguishable given only its input, i.e. without the secret values from the honest parties.

For (m_1, m_2), $\mathcal{A}$ can further decrypt m_2 using the private keys of $\mathbb{W}_1$ and $\mathbb{W}_2$, leaving the value only encrypted with g^{r_2z}. Both g^{r_2} and g^z are known to $\mathcal{A}$; however, according to the DDH assumption [32], for the triplet (g^{r_2}, g^z, g^{r_2z}), $\mathcal{A}$ is not able to distinguish g^{r_2z} from a random element, which is generated by Sim. In addition, for $\{m_3, m_4\}$, given the output of $f_v(\cdot)$, Sim can construct a simulated set $\mathbb{S}$ by drawing two random elements (or, drawing one random element if $f_v(\cdot) = \mathsf{false}$). The adversary $\mathcal{A}$ is not able to distinguish $\mathbb{S}$ from the real set $\{m_3, m_4\}$, except with negligible probability. Therefore, the simulator Sim is able to output messages that are indistinguishable given only its input. To formally present the view of adversary $\mathcal{A}$, we have $\{\mathsf{Sim}(I, g^z, a', x, y, x', y', f_v(a', b'))\} \overset{c}{\equiv} \{View_{\mathcal{A}}^{\Pi_v}(I, a', b', x, y, z, x', y', z')\}$.

The other two cases where worker $\mathbb{W}_1$ and worker $\mathbb{W}_2$ respectively remains honest follow a similar simulation algorithm, and hence we omit them here in the proof. □

Let n represent the average number of values per party from a total of m parties, and let w workers form the multiplicative deterministic re-encryption decryption mixnet. The communication cost in the mixnet is $O(nmw)$, and the computation cost is $O(nmw)$ in terms of modular exponentiation.

The proposed public-key based multiplicative deterministic re-encryption decryption mixnet provides a powerful protocol for many useful computations over encrypted data. This includes: 1) equality comparison directly on encrypted data, which is one fundamental primitive in numerous applications. If two values are equal, then their encrypted values are also equal. We showed this when computing PSI-CA in § 5.2.2; 2) set union and intersection cardinalities. For instance, when $\mathbb{P}_1$ and $\mathbb{P}_2$ encrypt a set of values, this re-encryption mixnet can securely compute set union cardinality and intersection cardinality; 3) homomorphic multiplication and division operations. For example, $a^{x'y'z'} \times b^{x'y'z'} = (a \times b)^{x'y'z'}$.

The multiplicative deterministic re-encryption decryption mixnet can be easily extended as well. For example, after the process shown in Figure 5.5, the workers can initiate a second round of decryption-only process, to decrypt the value $(a \times b)^{x'y'z'}$ for multiplicative operations, or the union/intersected elements for set union/intersection problems. We leave these straightforward extensions to readers for exercise.

5.3.2 Set-Level Equality Testing

The second mixnet is designed to achieve the following goal: given a set of input integers $\{v_1, \ldots, v_m\}$, where v_i is the confidential integer from party $\mathbb{P}_i$, and a boolean functionality $f_s(\cdot)$ taking input set and outputting $\texttt{true}$ if the sum of inputs is 0, $f_s(\cdot)$ can be securely fulfilled.

We now present the second mixnet, called *additive deterministic re-encryption decryption mixnet* to achieve above set-level equality testing. An additive mixnet shares many properties with multiplicative mixnet, and is more efficient for additive operations.

Figure 5.6 shows an example of such additive deterministic re-encryption decryption mixnet using three workers and two parties. We highlight the differences from the multiplicative deterministic re-encryption decryption mixnet (§ 5.3.1): 1) instead of encrypting an integer directly, each party encrypts its value v to g^v; 2) worker $\mathbb{W}_1$ multiplies the encrypted values from parties. This step constructs the sum of input values (on the exponent); 3) each worker goes through an re-encryption process for the additive value, and eventually the last worker gets 1 if the sum equals to 0.

Let Π_s denote an additive deterministic re-encryption decryption mixnet, we can derive the following theorem:

Theorem 12. Π_s *securely implements the functionality $f_s(\cdot)$ in the presence of semi-honest adversary given at least one party and one worker are not controlled by the adversary.*

Proof. The last worker fulfils the functionality $f_s(\cdot)$. If the sum of set is zero, then the encrypted value equals to one. Consider a simulator Sim, input information $I = \{\mathbb{G}_p, H(\cdot)\}$ a worst case scenario where the adversary $\mathcal{A}$ controls $\mathbb{P}_1$, workers $\mathbb{W}_1$ and $\mathbb{W}_2$ using Figure 5.6 as the example. The actual view of $\mathcal{A}$ consists of three messages m_{1-3}. In Figure 5.6, m_1 and m_2 correspond to the encrypted value $(g^{r_2}, g^b g^{r_2 x} g^{r_2 y} g^{r_2 z})$ from honest $\mathbb{P}_2$, and m_3 corresponds to the re-encrypted value $g^{(a+b)x'y'z'}$ from honest worker $\mathbb{W}_3$ (step 3.2). Similar to the proof for Theorem 11, $\mathcal{A}$ is not able to distinguish (m_1, m_2) because of DDH assumption. As for m_3, given the output of $f_s(\cdot)$, Sim can output a random value (or, 1 if $f_s(\cdot) = \texttt{true}$). The adversary cannot distinguish this from m_3, since x', y', and z' are freshly chosen for each run.

Therefore, the simulator Sim is able to generate messages that are indistinguishable given only its input. To form the view of adversary $\mathcal{A}$, we have $\{\mathsf{Sim}(I, g^z, a', x, y, x', y', f_s(a', b'))\} \overset{c}{\equiv} \{View_{\mathcal{A}}^{\Pi_s}(I, a', b', x, y, z, x', y', z')\}$. □

	Party $\mathbb{P}_1$	Party $\mathbb{P}_2$
0.1, Sync $K\{g^x, g^y, g^z\}$	$value : a = H(a')$	$value : b = H(b')$
0.2, Encrypt(g^{value}, K)	$(g^{r_1}, g^a g^{r_1 x} g^{r_1 y} g^{r_1 z})$	$(g^{r_2}, g^b g^{r_2 x} g^{r_2 y} g^{r_2 z})$
0.3, Send values to mixnet	$\Downarrow$	
	Worker $\mathbb{W}_1$. Keys: (x, g^x)	
1.0, Initialize	secret re-encryption key: $x' \leftarrow_{\$} \mathbb{Z}_p$	
1.1, Multiply	let $r = r_1 + r_2, (g^r, g^{a+b} g^{rx} g^{ry} g^{rz})$	
1.2, Decrypt	$(g^r, g^{a+b} g^{ry} g^{rz})$	
1.3, Re-encrypt	$(g^{rx'}, g^{(a+b)x'} g^{rx'(y+z)})$	
1.4, Re-randomize	$s_1 \leftarrow_{\$} \mathbb{Z}_p, g^{rx'+s_1}, g^{(a+b)x'} g^{(rx'+s_1)(y+z)})$	
1.5, Send to next	$\Downarrow$	
	Worker $\mathbb{W}_2$. Keys: (y, g^y)	
2.0, Initialize	secret re-encryption key: $y' \leftarrow_{\$} \mathbb{Z}_p$	
2.1, Decrypt	let $m_1 = rx' + s_1, (g^{m_1}, g^{(a+b)x'} g^{m_1 z})$	
2.2, Re-encrypt	$(g^{m_1 y'}, g^{(a+b)x'y'} g^{m_1 y' z})$	
2.3, Re-randomize	$s_2 \leftarrow_{\$} \mathbb{Z}_p, (g^{m_1 y' + s_2}, g^{(a+b)x'y'} g^{(m_1 y' + s_2)z})$	
2.4, Send to next	$\Downarrow$	
	Worker $\mathbb{W}_3$. Keys: (z, g^z)	
3.0, Initialize	secret re-encryption key: $z' \leftarrow_{\$} \mathbb{Z}_p$	
3.1, Decrypt	let $m_2 = m_1 y' + s_2, (g^{m_2}, g^{(a+b)x'y'})$	
3.2, Re-encrypt	$g^{(a+b)x'y'z'}$	

Figure 5.6: An additive, deterministic re-encryption decryption mixnet with 3 workers and 2 parties.

Similar to the multiplicative deterministic re-encryption decryption mixnet, the additive deterministic re-encryption decryption mixnet also has a linear computation and communication cost. For a setting with m parties and w workers, both the computation and communication cost are $O(m + w)$ in terms of modular exponentiation.

The proposed additive deterministic re-encryption decryption mixnet can efficiently empower a category of additive comparisons. This includes: 1) value-to-value equality check. To compare whether $a = b$, we can construct a set of $\{a, -b\}$ as the input to the mixnet. If the encrypted output is 1, then it implies $a = b$; 2) value-to-set evaluation. The mixnet provides a way to evaluate the sum of the set to a value t. Consider a set of size n, the input $n+1$ values (n values from the set in addition to $-t$) can be evaluated using the mixnet; 3) set-to-set sum comparison. Comparing the equality of the sum of two sets can be securely implemented by adding all values from one set and all inverted value from the other.

5.3.3 Parallelizing MixNet

While the mixnets incur linear costs, they can be further scaled out to multiple chains for performance improvement.

Consider two chains $\mathbb{C}_1$ and $\mathbb{C}_2$, each of them consists of w_1 and w_2 workers respectively. w_1 and w_2 are not necessarily to be equal, but for simplicity, we assume two chains are of same length. For the multiplicative deterministic re-encryption decryption mixnet, continuing the example in Figure 5.5, $\mathbb{C}_1$ consists of 3 workers with secret re-encryption keys x'_1, y'_1 and z'_1 respectively. Let $\mathbb{C}_2$ have 3 workers with secret re-encryption keys x'_2, y'_2 and z'_2. Let $\mathbb{P}_1$ go with $\mathbb{C}_1$, and $\mathbb{P}_2$ go with $\mathbb{C}_2$. The output of the two chains would be $a^{x'_1 y'_1 z'_1}$ and $b^{x'_2 y'_2 z'_2}$. If the exponents are equivalent (i.e., $x'_1 y'_1 z'_1 = x'_2 y'_2 z'_2$), then the equality check of input values does work across different chains. In the following, we describe the pre-processing model to generate secret re-encryption keys.

Consider a trusted oracle $\mathcal{O}$ and a cyclic group $\mathbb{Z}_p$. At the beginning, $\mathcal{O}$ randomly fix a value $M \in \mathbb{Z}_p$. Given w, which is the length of a chain $\mathbb{C}$, $\mathcal{O}$ randomly samples $w-1$ values $s_1, ..., s_{w-1} \in \mathbb{Z}_p$, and sets $s_w = M \times (s_1 \times ... \times s_{w-1})^{-1}$. s_i corresponds to the secret re-encryption key on worker $\mathbb{W}_i$. $\mathcal{O}$ sends s_i to worker $\mathbb{W}_i$ via a secure channel. $\mathcal{O}$ reuses M when it needs to generate a new set of keys. Obviously an value v is always encrypted to v^M regardless of chains.

The above process also works for the additive deterministic re-encryption decryption mixnet. In practice, $\mathcal{O}$ can be easily implemented using secure computations [26].

5.4 Secure Congenial FD Discovery

So far, we adopted the traditional definition of FD (Definition 1) in the secure multi-party settings, and required the discovered FDs to hold on the whole dataset. In this section, we relax that requirement; we define f_c as an FD that holds on any of the partition. Note that f_c might not be an valid FD on the whole dataset. For example, in Figure 5.1, f_2 is valid on both D_1 and D_2, but is not an valid FD on D. Formally, let $\sum_c$ represent the set of all non-trivial and minimal FDs that are valid on every partition $D_i(i \in [1, m])$: $\sum_c \models \hat{S}$. For the purpose of distinction, we call an FD $f \in \sum$ the union FD (uFD, in short), and an FD $f_c \in \sum_c$ the congenial FD (cFD, in short).

To contrast the utility of cFD with uFD, consider the following: after learning the set of uFDs, each party learns two pieces of profiling information: 1) the set of uFDs, which holds globally on the entire dataset; and 2) all the local FDs that are invalid on other partitions, because they were not part of uFDs. On the other hand, by learning the set of cFDs, each party will know: 1) a superset of uFDs; and 2) a subset of local FDs that is invalid on other partitions. Both pieces are useful profiling information (especially the pruned local FDs). Another way to leverage cFD is in pruning the search space of the FD discovery algorithm (§ 5.1.3): $f \in \sum$ is true either $f \in \sum_c$ or $\exists f_c \in \sum_c$ that $f \in Spec(f_c)$. All the parties do not need to validate a candidate FD f if it has no chance of being an uFD given the set of cFDs (e.g., if the left hand side attributes of f does not intersect with that of any $f_c \in \sum_c$).

We first propose the distributed cFD validation in § 5.4.1, and then present a voting-based protocol for secure cFD validation in § 5.4.2.

5.4.1 Distributed cFD Validation

In the case of cFD, rather than validating all data across partitions, discovering cFD only requires to validate the *agreement* by all parties. Intuitively, for a given candidate cFD, if every party agrees its validity on its data partition, then the candidate is a valid cFD.

In general, for each candidate cFD $f_c : A \to B$, a naïve way to validate it is that each party $\mathbb{P}_i$ can compute attribute errors $e(A^{D_i})$ and $e((AB)^{D_i})$ locally on their data, and determine the validity using Lemma 3 (§ 5.2.1). But this general approach would require to compute the attribute partitions every time for each validation. In order to avoid such cost, we propose that each party first computes its FD set S_i using its data partition, and then infers the validity of a given candidate cFD f_c using S_i. This approach is inspired by the following property of the cFD:

Property 1. *Given a set of m FD sets $\mathbb{S} = \{S_1, ..., S_m\}$, where S_i is the set of all minimal, non-trivial FDs on partition D_i. A congenial FD f_c on $\mathbb{S}$ satisfies that $\forall S \in \mathbb{S}, \exists f \in S, f_c \in Spec(f)$. A congenial FD f_c is minimal if $\nexists f'_c$ that $f_c \in Spec(f'_c)$ and f'_c is a congenial FD.*

Example 11: Consider the following FDs in Figure 5.1:

Algorithm 16 cFD inference

```
Input: $f : A \to B$                          ▷ candidate cFD to be evaluated
Input: $S$                  ▷ set of all minimal non-trivial FDs of the party
 1: procedure INFER($f, S$)
 2:     for all $A' \to B \in S$ do
 3:         $A^* = A \cup A'$
 4:         if $A^* \equiv A$ then                    ▷ check specialization
 5:             return 1
 6:         end if
 7:     end for
 8:     return 0
 9: end procedure
```

$f_3 : occupation \to income \qquad f_4 : martial \to income$

$f_5 : occupation, marital \to income$

By examining D_1, $f_3 \in S_1$, and $f_5 \in Spec(f_3)$. Similarly on D_2, $f_4 \in S_2$, and $f_5 \in Spec(f_4)$. Hence, f_5 in a congenial FD. Furthermore, f_5 is also minimal because neither f_3 nor f_4 is an valid congenial FD. □

Algorithm 16 utilizes Property 1 to do inference using a party's FDs only. The idea is to check whether the candidate cFD can be specialized by any of its FDs which share the same right-hand side attribute B. If the candidate cFD can be specialized by all the parties, then the candidate is a true cFD. This leads to the solution for securely verifying candidate cFDs: every party securely votes either 1 if the candidate is a valid FD on its data partition, or 0 if not; and then we can construct an additive deterministic re-encryption decryption mixnet to securely check whether the sum of all votes is equal to the total number of parties.

5.4.2 A Secure cFD Discovery Protocol

The prerequisite in cFD protocol is to discover the set of all minimal and non-trivial FDs S_i by each party from its data D_i. This can be done locally by every party running an existing FD discovery algorithm, e.g., [86]. Similarly to the uFD discovery protocol in § 5.2.2, we arbitrarily choose a moderator to initialize the search space and drive pruning. To prune the search space (§ 5.1.3), Figure 5.7 shows such a protocol to implement $CheckFD(f_c)$ for a given candidate cFD f_c. First, the moderator instructs all parties to vote either 1

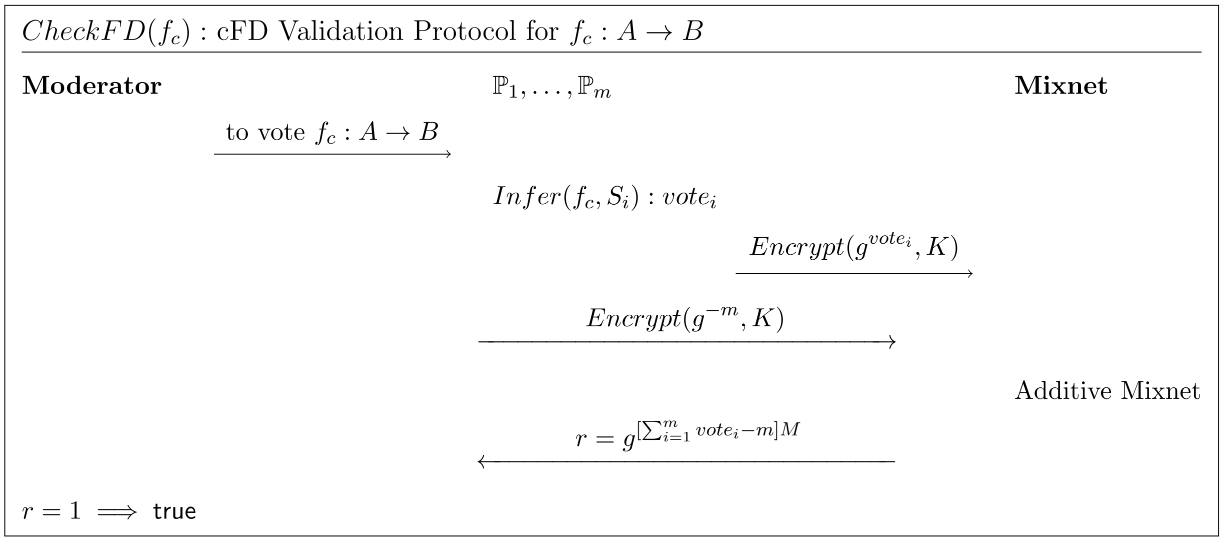

$CheckFD(f_c)$: cFD Validation Protocol for $f_c : A \to B$
Moderator
$\mathbb{P}_1, \ldots, \mathbb{P}_m$
Mixnet
to vote $f_c : A \to B$
$Infer(f_c, S_i) : vote_i$
$Encrypt(g^{vote_i}, K)$
$Encrypt(g^{-m}, K)$
Additive Mixnet
$r = g^{[\sum_{i=1}^{m} vote_i - m]M}$
$r = 1 \implies$ true

Figure 5.7: The cFD validation protocol for securely validating a candidate cFD using mixnet.

if the candidate is valid on their data, or 0 otherwise. Each party runs Algorithm 16 on the candidate f_c and their own FD set S_i, and sends its encrypted vote to an additive deterministic re-encryption decryption mixnet (§ 5.3.2). In addition, the encrypted inverse of the number of parties (i.e., $-m$) is also sent as an input. The output r of the mixnet is an encrypted value, indicating whether all parties voted 1 or not. If $r = 1$, then all parties have voted 1, meaning f_c is a true cFD.

5.5 Evaluation

5.5.1 Experiment Setup

Datasets. We choose 8 datasets from the UCI ML repository [49]. These datasets have different number of columns, rows and FDs. The details are listed in Table 5.1.

To simulate the data partitions that are held on parties, we horizontally chunk the datasets into m disjoint and even parts where m is the number of parties. For reproducibility purposes, a scale factor SF is used to represent the number of chunks that each partition has. Every partition is composed of SF number of consecutive chunks in a round-robin sequence. By default, SF is 1 for all datasets. Although data size is evenly distributed, the values of attribute partitions (used in uFD discovery) and local FDs (used in cFD discovery) can have skewness.

Metrics. We measure two end-to-end costs: 1) computation cost, measured by the overall FD discovery time; 2) communication cost, counted by the bytes which are transferred through all the nodes. Every setting reports the mean and standard error of 3 runs.

Implementation details. We conducted all experiments on AWS platform. Each party or worker was deployed on an EC2 instance locating in spread networks across us-west regions. Each instance has 72 vCPU and 144 GB memory. The solutions are implemented in C++ with gcc-5.4.0, and run on the Ubuntu-14.04.

Security parameters are fixed as follows: both the length of generator g in ElGamal and the length of keys are set to be 256 bits. p has a length of 3072 bits.

In additional, we also compared our solution (tagged by *SMFD*) with distributed FD validation using two general purpose MPC protocols: 1) *MultipartyPSI* [104], for multi-party private set intersection. We integrate their code[2] by replacing the mixnet with set intersections among parties. Specifically, for uFD, we compute the set intersections in Eqn. (5.3). Due to the limitation that MultipartyPSI requires all parties have same-sized sets, we assign each party to own the full dataset to avoid the overhead of padding. cFD protocol is implemented by intersecting votes of all parties. 2) *SMCQL* [23], for secure SQL querying over the union of its source databases, and serves as a general approach to discover uFDs. Due to the limitation of the current implementation[3], we compute the attribute partition errors $e(A)$ and $e(AB)$ by issuing two SQL queries to select `count(distinct A)` and `count(distinct AB)`, respectively. SMCQL was tested with two databases on the same host. SMFD's code, data and evaluation metrics are open sourced on GitHub: `https://github.com/cgebest/smfd`.

5.5.2 Overhead of SMFD

Table 5.1 shows the end to end cost comparison between the proposed SMFD and the non-secure plaintext-based distributed FD validation, in a setting of 3 parties, 3 workers, where each party or worker node uses 128 threads for public key operations. The security

[2]`https://github.com/osu-crypto/MultipartyPSI`

[3]`https://github.com/smcql/smcql`

[4]based on the partition strategy in § 5.5.1.

Table 5.1: End to end comparison between SMFD and the plaintext-based distributed FD validation

				Comput. (Sec)			Comm. (MB)				Comput. (Sec)			Comm. (MB)		
Dataset	# of Col	# of Row	# of uFD	SM FD	Plain Text	Over head	SM FD	Plain Text	Over head	# of cFD[4]	SM FD	Plain Text	Over head	SM FD	Plain Text	Over head
iris	5	150	4	7.6	0.5	15	9.8	0.1	98	4	3.0	0.3	10	0.4	0.01	40
balance-scale	5	625	1	9.0	0.5	18	13.8	0.1	138	1	3.5	0.4	9	0.5	0.01	50
chess	7	28056	1	420.5	2.9	143	1304.9	11.5	113	1	20.4	2.0	10	2.7	0.06	45
abalone	9	4177	137	1819.5	11.6	157	5644.7	104.7	54	159	64.1	6.2	10	8.4	0.19	44
nursery	9	12960	1	532.5	31.5	17	1333.6	65.8	20	9	94.6	9.1	10	12.4	0.29	43
breast-cancer	11	699	46	1508.1	98.8	15	3190.6	35.4	90	77	421.0	69.5	6	55.0	1.36	40
bridges	13	108	142	1726.8	197.8	9	2130.0	47.2	45	127	753.0	130.0	6	98.6	2.53	39
echocardiogram	13	132	538	1342.3	137.0	10	1958.0	30.5	64	420	506.2	92.3	5	66.2	1.70	39

overhead using all datasets is often within one order of magnitude for both uFD and cFD discoveries, for both computation and communication costs.

The computation overhead stems from public key operations, and the communication overhead is due to ciphertext expansion. As Table 5.1 shows, FD discovery costs are highly data dependent. The overall cost is dominated by two factors: 1) the cost of validating one FD, and 2) the number of FDs that require validation. The first factor is determined by the size of attribute partition values for uFD, which usually but not necessarily increases with the number of rows; while for cFD, the cost is independent because we process only votes from parties. Meanwhile, the second factor is determined by the number of columns and FDs. Larger number of columns lead to more complex lattice with more FD candidates; while a true FD can prune out all its specialization from the lattice.

The overhead of discovering uFD is usually heavier than cFD, because in uFD discovery, the values of attribute partitions pass through the mixnet, and number of values can be large. Therefore, the computation cost of uFD can be reduced by using multiple threads, where multiple threads can work on values in parallel.

To show the benefit of parallel processing, Figure 5.8 illustrates the uFD computation overhead using different threads per party and worker node. Using more threads reduces the overhead, implying that our solution can benefit more from modern hardware. Even using 8 threads, the overhead is still often within one order of magnitude. The overhead can be further reduced by scaling out to multiple mixnet chains (§ 5.3.3) and here we omit because of similar behaviour. In the rest of experiments, we show results using one thread per node and one mixnet chain.

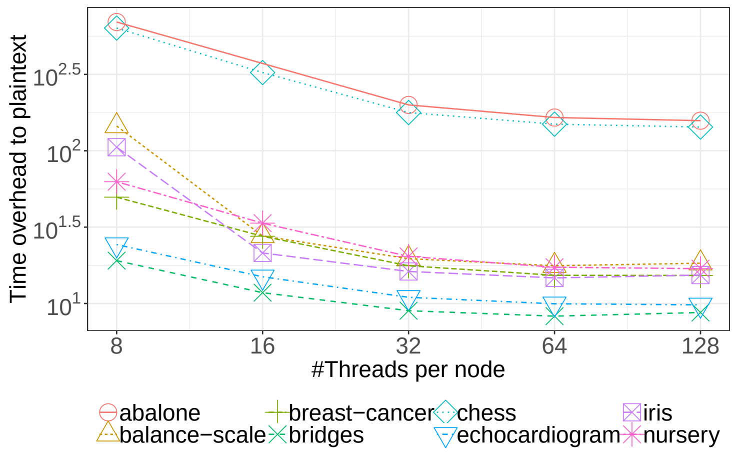


Figure 5.8: uFD computation overhead to the plaintext decreases with more threads per instance.

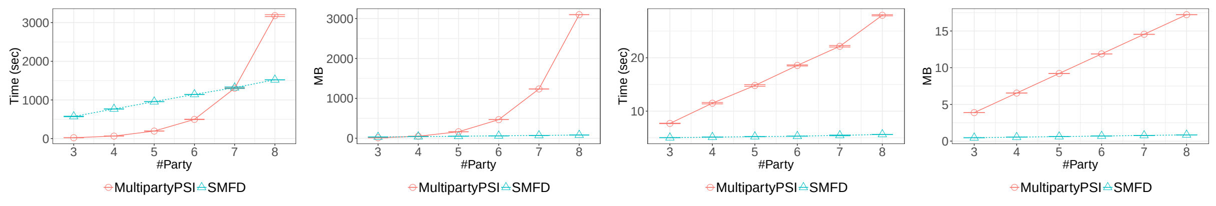
5.5.3 Efficiency of SMFD

Comparison to MultipartyPSI. We compare SMFD with distributed FD vaidation using MultipartyPSI [104]. Note that MultipartyPSI has a weaker security model since parties learn the intersected values.

Figure 5.9a and Figure 5.9b illustrate the costs for discovering uFD. Recall § 5.2.1 that for each uFD validation, there are $2^m - m - 1$ intersections for m parties. Although one set intersection using MultipartyPSI is cheaper because of the use of symmetric key cryptography instead of public key cryptography, the exponential number of sets overwhelms the overall performance. For example, in Figure 5.9a, when the number of parties is small, MultipartyPSI runs faster. With more parties, the benefit of faster processing per intersection is soon diminished by the exponential number of intersections. When there are 8 or more parties, SMFD starts to outperform MultipartyPSI. In contrast, SMFD costs increase since more parties are sending data to the mixnet; however, computing PSI-CA only incurs linear complexity (§ 5.2.2).

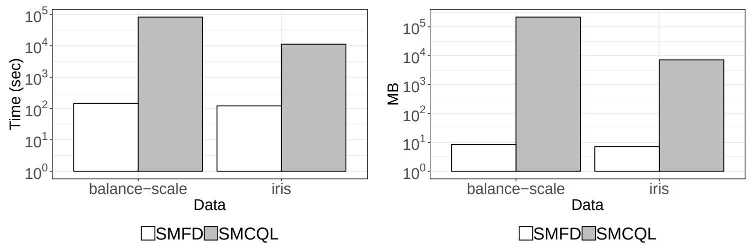
Figure 5.9c and Figure 5.9d show the costs of discovering uFD. MultipartyPSI requires larger costs than SMFD and is more sensitive to the number of parties.

Comparison to SMCQL. Figure 5.10a compares the execution time of discovering uFDs. On balance-scale dataset, SMCQL takes more than 22 hours to finish validating all uFDs, while SMFD completes in 146 seconds. On iris dataset, SMCQL takes 3.2 hours, and SMFD finishes in 121 seconds. On both datasets, SMFD is two orders of magnitude faster.



(a) Comp. Cost uFD (b) Comm. Cost uFD (c) Comp. Cost cFD (d) Comm. Cost cFD

Figure 5.9: Cost comparison between SMFD and distributed FD discovery using MultipartyPSI, with increasing number of parties on the balance-scale dataset: a) and b) show the computation and communication cost for uFD, respectively. SMFD has a linearly cost, while MultipartyPSI incurs an exponential cost. c) and d) show the costs for cFD. MultipartyPSI requires larger costs and is more sensitive to the number of parties.



(a) Computation Cost (b) Communication Cost

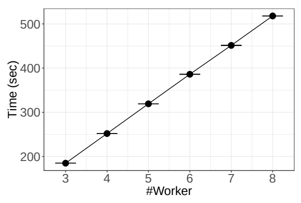
Figure 5.10: Cost comparison between SMFD and SMCQL.

Figure 5.10b shows the comparison of communication cost. SMCQL intensively inserts dummy tuples to pad intermediate results during its execution, so it is expected that communication cost is expensive. On balance-scale dataset, SMFD costs 8MB, while SMCQL transfers more than 200GB. On iris dataset, SMFD consumes 7MB, outperforming SMCQL's 7178MB by three orders of magnitude.

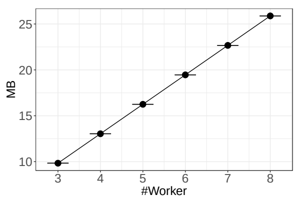
Note that SMCQL was tested on the same host, and the difference of computation cost in distributed scenario is expected to be amplified by its communication cost.

5.5.4 Scalability of SMFD

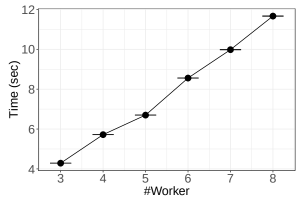
Scalability of varying the number of workers. Figure 5.11 shows the costs of varying the number of workers. The costs of computation and communication increase linearly with the number of workers, for both types of FDs.



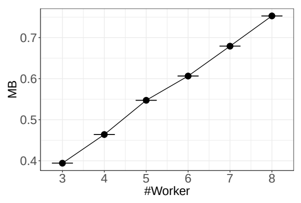
(a) Comp. Cost uFD



(b) Comm. Cost uFD

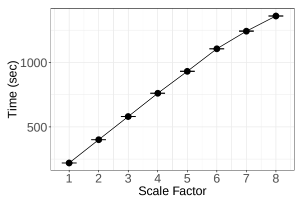


(c) Comp. Cost cFD

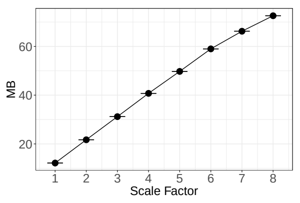


(d) Comm. Cost cFD

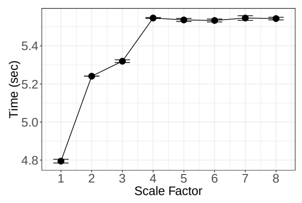
Figure 5.11: Cost of varying the number of workers using 3 parties on the iris dataset.



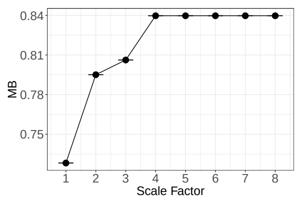
(a) Comp. Cost uFD



(b) Comm. Cost uFD



(c) Comp. Cost cFD



(d) Comm. Cost cFD

Figure 5.12: Cost of varying the data volume using 8 parties and 3 workers on the iris dataset.

Adding more workers does not affect FD candidates, but only affects the performance of mixnet. Recall that in a mixnet chain (§ 5.3), all workers collaborate in a sequential way, and they share the same type of job. Hence, the cost increases linearly with the number of workers.

Scalability of varying data volume. Figure 5.12 measures the cost of changing data volume per party. Figure 5.12a and Figure 5.12b show that costs increase with SF for discovering uFDs. Generally, the larger data volume on each party, the larger the attribute partition set is expected. Therefore, both the execution time and transferred bytes increase.

However, uFD highly correlates with the FDs that are discovered by each party, and the size of candidate uFD is not static. When SF is small, increasing the data volume per party invalidates more uFDs, and consequently triggers more inference (Algorithm 16). When $SF > 4$, increasing data volume per party does not provide more insights, and hence it is expected the cost remains the same. Figure 5.12c and Figure 5.12d illustrate such trend.

5.6 Conclusion

This chapter focuses on discovering FDs in the secure multi-party scenario against semi-honest adversaries. We formulate discovering FDs, design secure constructions for FD validation, and present efficient protocols to enable secure multi-party FD discovery. Our experimental results show the linear scalability of our protocols and over 2 orders of magnitude performance benefit compared to the general purpose secure multi-party computation.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this dissertation, we focus on enabling data analytics on private siloed data and solved the technical challenges in three specific analytics tasks when interacting with private datasets.

The first task is accuracy-aware differentially private data exploration. Current systems for answering queries with differential privacy place an inordinate burden on the data scientist to understand differential privacy, manage their privacy budget, and even implement new algorithms for noisy query answering. Moreover, current systems do not provide any guarantees to the data scientist on the quality they care about, namely accuracy of query answers. To solve the problem, we propose APEx, a novel system that allows data scientist to pose adaptively chosen sequences of queries along with required accuracy bounds. By translating queries and accuracy bounds into differentially private algorithms with the least privacy loss, APEx returns query answers to the data scientist that meet the accuracy bounds, and proves to the data owner that the entire data exploration process is differentially private. Our comprehensive experimental study on real datasets demonstrates that APEx can answer a variety of queries accurately with moderate to small privacy loss, and can support data exploration for entity resolution with high accuracy under reasonable privacy settings.

The second task is constraint-aware differentially private data synthesis. 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. As a result, the synthesized data is not useful for any downstream tasks that require this structure to be preserved. To solve the problem, we propose 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.

The third task is secure and efficient functional dependency (FD) discovery over multi-parties. Discovering functional dependencies usually requires access to all data partitions to find constraints that hold on the whole dataset. Simply applying general secure multi-party computation protocols incurs high computation and communication cost. In SMFD, we formulate the FD discovery problem in the secure multi-party scenario. We propose secure constructions for validating candidate FDs, and present efficient cryptographic protocols to discover FDs over distributed partitions. Experimental results show that solution is practically efficient over non-secure distributed FD discovery, and can significantly outperform general purpose multi-party computation frameworks.

6.2 Future Work

The increasing concern of data privacy poses significant challenges to utilize private data assets. This dissertation provides first a few steps toward *enabling private data science*, where data privacy constraint is automatically and holistically integrated into data science pipelines. Our approach to this vision follows the similar pattern as that in this thesis, which starts with understanding the challenges in each data science task, and then designs novel solutions to solve specific problems. Three of which we believe as natural extensions to this dissertation are the following.

Private Data Preparation. Data preparation is the process of manipulating raw data into a consumption-ready form, and it involves many tasks such as data integration, data cleaning, data augmentation and data wrangling. Data preparation is essential as a prerequisite for data-driven applications. Conducting data preparation is time-consuming and heavily requires human-in-the-loop, such as to write rules to extract the correct information, identify transformations to normalize values and provide domain knowledge to match schema. There has been an on-going effort to achieve self-serve data preparation, and recent works adopt machine learning to automate the data preparation process. However, those machine learning based solutions are data-hungry and still need human involvement to label and annotate training data, which becomes problematic when data have sensitive information. Automating private data preparation presents an exciting spectrum of research opportunities, and we plan to start with the following directions. First, data annotation is a complex subjective process and is often difficult to form into explicit queries or data properties. Based on the techniques that we developed for private data access via privacy query engine and synthetic data, we plan to investigate an useful access interface layer for private data annotation. Second, considering the open availability of public information, such as public datasets and pre-trained language models, it would be interesting to leverage the prior information, which can be useful and incurs no privacy cost, to aid private data preparation.

Private Workload Synthesis. Many of the design choices of database architecture such as storage layout and index structures involve tuning by accessing a representative set of data and query (a.k.a., workload). Despite the long history of research on automating this tuning process, such as using rule-based models to find the best parameters and more recently applying reinforcement learning to learn instance-optimized components, there is still no turn-key solution to fit all scenarios with different combinations of data, workload and hardware. Therefore, DBAs and cloud database vendors often need to access the data and workload to tune database architecture and explain decisions. In particular, database workload can breach data privacy in various ways: 1) a SQL query itself may have sensitive literals; 2) the choice of operators in a query plan reflects certain data statistics (e.g., using a loop join in the query plan implies high selectivity for the query condition.); and 3) the cardinality of each operator in a query plan directly reveals the statistics of the private data. When sensitive data and workload prevent direct access for applying traditional tuning practices, database systems are likely to suffer from suboptimal performance. From the techniques that we designed for private data synthesis, we would like to expand to synthesize private workload for tuning data systems.

Federated Knowledge Graph Construction. Data silos, or broadly interpreted, knowledge silos widely exist due to security and privacy constraints. Based on the techniques that we developed for federated data profiling, I would like to tackle the construction of federated knowledge graphs. A knowledge graph (KG) is a heterogeneous graph composed of nodes as entities and edges as relations, and knowledge is represented as a factual triple of entities connected by a relation. Ubiquitous KGs have been shown to be effective in facilitating search, mining and other complex data-driven tasks. Therefore, federating KGs becomes critical to support rich data services. Federated KG construction under security and privacy guarantee presents a wide range of open challenges and opportunities. In particular, we plan to tackle two critical aspects as follows. First, the privacy constraint on knowledge silos requires redesigning the full stack of privacy-preserving KG tasks, involving knowledge extraction, representation, integration, as well as query answering. We intend to develop innovative algorithms and scalable systems for each of the KG tasks. Second, in contrast to traditional KG evaluation, which has benefited from open and rich set of data sources such as Wikidata and DBpedia, due to the sensitive nature of knowledge silos, they are often sparse and have limited access for debugging and evaluation. Based on the techniques that we developed for synthesizing relational data, investigating synthetic KGs to facilitate the evaluation and development of federated KG construction is an exciting future work.

References

[1] Amazon gets record $888 million eu fine over data violations. https://www.bloomberg.com/news/articles/2021-07-30/amazon-given-record-888-million-eu-fine-for-data-privacy-breach. Accessed: 2022-03-13.

[2] British airways fined £20m for data breach affecting 400,000 customers. https://www.telegraph.co.uk/technology/2020/10/16/ico-fines-british-airways-20m-data-breach/. Accessed: 2022-03-13.

[3] Companies need to take responsibility for protecting sensitive user data. https://www.entrepreneur.com/article/242355. Accessed: 2022-03-13.

[4] Equifax agrees to settlement of up to $700 million over 2017 data breach. https://www.theverge.com/2019/7/22/20703497/equifax-ftc-fine-settlement-2017-data-breach-compensation-fund. Accessed: 2022-03-13.

[5] Tlc trip record data. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.

[6] Yahoo says hackers stole data on 500 million users in 2014. https://www.nytimes.com/2016/09/23/technology/yahoo-hackers.html. Accessed: 2022-03-13.

[7] 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.

[8] scikit-learn, machine learning in python, Version 0.23.2.

[9] The tpc benchmark h (tpc-h), Version 2.18.0.

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[11] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, pages 308–318. ACM, 2016.

[12] Ziawasch Abedjan, Patrick Schulze, and Felix Naumann. DFD: efficient functional dependency discovery. In *CIKM*, pages 949–958, 2014.

[13] John M. Abowd. The U.S. census bureau adopts differential privacy. In *KDD*, page 2867, 2018.

[14] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *EuroSys*, pages 29–42. ACM, 2013.

[15] David W. Archer, Dan Bogdanov, Yehuda Lindell, Liina Kamm, Kurt Nielsen, Jakob Illeborg Pagter, Nigel P. Smart, and Rebecca N. Wright. From keys to databases - real-world applications of secure multi-party computation. *Comput. J.*, 61(12):1749–1771, 2018.

[16] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017.

[17] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory Comput.*, 8(1):121–164, 2012.

[18] Brooke Auxier, Lee Rainie, Monica Anderson, Andrew Perrin, Madhu Kumar, and Erica Turner. Americans and privacy - concerned confused and feeling lack of control over their personal information. *Pew Research Center*, 2019.

[19] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

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

[21] Raef Bassily, Adam Groce, Jonathan Katz, and Adam D. Smith. Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy. In *FOCS*, pages 439–448. IEEE Computer Society, 2013.

[22] Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. FOCS. pages 464–473. IEEE Computer Society, 2014.

[23] Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel N. Kho, and Jennie Rogers. SMCQL: secure query processing for private data networks. *PVLDB*, 10(6):673–684, 2017.

[24] Johes Bater, Xi He, William Ehrich, Ashwin Machanavajjhala, and Jennie Rogers. Shrinkwrap: Efficient sql query processing in differentially private data federations. *PVLDB*, 12(3), November 2018.

[25] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *CCS*, pages 784–796. ACM, 2012.

[26] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[27] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012.

[28] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006.

[29] Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition.* Information science and statistics. Springer, 2007.

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

[31] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 224–241. Springer, 2009.

[32] Dan Boneh. The decision diffie-hellman problem. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 48–63, 1998.

[33] Claire McKay Bowen and Fang Liu. Comparative study of differentially private data synthesis methods. *Statistical Science*, 35(2):280–307, May 2020.

[34] Meta S Brown and S Kudyba. Transforming unstructured data into useful information. In *Big Data, Mining, and Analytics: Components of Strategic Decision Making*. Taylor & Francis, 2014.

[35] U.S. Census Bureau. Lehd origin-destination employment statistics (2002-2017), Accessed on 2020-11-30.

[36] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

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

[38] Qingrong Chen, Chong Xiang, Minhui Xue, Bo Li, Nikita Borisov, Dali Kaafar, and Haojin Zhu. Differentially private data generative models. *CoRR*, abs/1812.02274, 2018.

[39] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. Differentially private high-dimensional data publication via sampling-based inference. In *SIGKDD*, pages 129–138, 2015.

[40] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD*, pages 785–794. ACM, 2016.

[41] Fei Chiang and Dhruv Gairola. Infoclean: Protecting sensitive information in data cleaning. *ACM J. Data Inf. Qual.*, 9(4):22:1–22:26, 2018.

[42] David Maxwell Chickering. Learning bayesian networks is np-complete. In *AISTATS*, pages 121–130. Springer, 1995.

[43] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. Discovering denial constraints. *PVLDB*, 6(13):1498–1509, 2013.

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

[45] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptol. ePrint Arch.*, page 86, 2016.

[46] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In *CANS*, pages 218–231, 2012.

[47] Rachel Cummings, Sara Krehbiel, Kevin A. Lai, and Uthaipon Tao Tantipongpipat. Differential privacy for growing databases. In *NeurIPS*, pages 8878–8887, 2018.

[48] Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, and Pradap Konda. The magellan data repository. https://sites.google.com/site/anhaidgroup/projects/data.

[49] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[50] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[51] Cynthia Dwork. Differential privacy. In *ICALP*, volume 4052, pages 1–12. Springer, 2006.

[52] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, volume 4004, pages 486–503. Springer, 2006.

[53] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[54] Cynthia Dwork and Moni Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *J. Priv. Confidentiality*, 2(1), 2010.

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

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

[57] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. Distributed representations of tuples for entity resolution. *Proc. VLDB Endow.*, 11(11):1454–1467, 2018.

[58] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 2007.

[59] David W. Embley. Key. In *Encyclopedia of Database Systems, Second Edition*. Springer, 2018.

[60] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067. ACM, 2014.

[61] Ju Fan, Tongyu Liu, Guoliang Li, Junyou Chen, Yuwei Shen, and Xiaoyong Du. Relational data synthesis using generative adversarial networks: A design space exploration. *Proc. VLDB Endow.*, 13(11):1962–1975, 2020.

[62] Liyue Fan. A survey of differentially private generative adversarial networks. In *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*, 2020.

[63] Peter A. Flach and Iztok Savnik. Database dependency discovery: A machine learning approach. *AI Commun.*, 12(3):139–160, August 1999.

[64] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *SEC*, pages 151–164, 2019.

[65] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. Dual query: Practical private query release for high dimensional data. In *ICML*, volume 32, pages 1170–1178, 2014.

[66] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

[67] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam D. Smith. Composition attacks and auxiliary information in data privacy. In *ACM SIGKDD*, pages 265–273. ACM, 2008.

[68] Chang Ge, Xi He, Ihab F. Ilyas, and Ashwin Machanavajjhala. APEx: Accuracy-aware differentially private data exploration. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD*, pages 177–194, 2019.

[69] Chang Ge, Ihab F. Ilyas, and Florian Kerschbaum. Secure multi-party functional dependency discovery. *Proc. VLDB Endow.*, 13(2):184–196, 2019.

[70] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F. Ilyas. Kamino: Constraint-aware differentially private data synthesis. *Proc. VLDB Endow.*, 14(10):1886–1899, 2021.

[71] O. Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *STOC*, pages 182–194, 1987.

[72] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications.* Cambridge University Press, 2004.

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

[74] Andy Greenberg. Apple's 'differential privacy' is about collecting your data—but not *your* data. *Wired*, 2016.

[75] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *NIPS*, pages 5767–5777, 2017.

[76] Rahul Gupta. Data augmentation for low resource sentiment analysis using generative adversarial networks. In *ICASSP*, pages 7380–7384. IEEE, 2019.

[77] Qilong Han, Qianqian Chen, Liguo Zhang, and Kejia Zhang. HRR: a data cleaning approach preserving local differential privacy. *Int. J. Distributed Sens. Networks*, 14(12), 2018.

[78] Samuel Haney, Ashwin Machanavajjhala, John M. Abowd, Matthew Graham, Mark Kutzbach, and Lars Vilhuber. Utility cost of formal privacy for releasing national employer-employee statistics. In *SIGMOD*, pages 1339–1354. ACM, 2017.

[79] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *NIPS*, pages 2348–2356, 2012.

[80] Michael B. Hawes. Implementing differential privacy: Seven lessons from the 2020 united states census. *Harvard Data Science Review*, 4 2020. https://hdsr.mitpress.mit.edu/pub/dgg03vo6.

[81] Michael Hay, Ashwin Machanavajjhala, Gerome Miklau, Yan Chen, and Dan Zhang. Principled evaluation of differentially private algorithms using dpbench. In *SIGMOD*, pages 139–154. ACM, 2016.

[82] Xi He, Ashwin Machanavajjhala, and Bolin Ding. Blowfish privacy: tuning privacy-utility trade-offs using policies. In *SIGMOD*, pages 1447–1458, 2014.

[83] Hans Hinterberger. Exploratory data analysis. In *Encyclopedia of Database Systems, Second Edition*. Springer, 2018.

[84] Justin Hsu, Aaron Roth, and Jonathan R. Ullman. Differential privacy for the analyst via private equilibrium computation. In *STOC*, pages 341–350. ACM, 2013.

[85] Yu Huang, Mostafa Milani, and Fei Chiang. Privacy-aware data cleaning-as-a-service. *Inf. Syst.*, 94:101608, 2020.

[86] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. TANE: an efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999.

[87] IBM. Cost of a data breach report. 2020.

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

[89] John Jablonski. Reputation damage control: Insuring the cost of mitigating reputational harm following a cyber-attack or data breach. *USLAW Magazine*, 2015.

[90] Anja Jerichow, Jan Müller, Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. Real-time mixes: a bandwidth - efficient anonymity protocol. *IEEE Journal on Selected Areas in Communications*, 16(4):495–509, 1998.

[91] Xiaoqian Jiang, Zhanglong Ji, Shuang Wang, Noman Mohammed, Samuel Cheng, and Lucila Ohno-Machado. Differential-private data publishing through component analysis. *Trans. Data Priv.*, 6(1):19–34, 2013.

[92] Aaron Johnson and Vitaly Shmatikov. Privacy-preserving data exploration in genome-wide association studies. In *SIGKDD*, pages 1079–1087. ACM, 2013.

[93] Noah M. Johnson, Joseph P. Near, and Dawn Song. Towards practical differential privacy for SQL queries. *Proc. VLDB Endow.*, 11(5):526–539, 2018.

[94] Theodore Johnson. Data profiling. In *Encyclopedia of Database Systems, Second Edition*. Springer, 2018.

[95] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. PATE-GAN: generating synthetic data with differential privacy guarantees. In *ICLR*, 2019.

[96] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021.

[97] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

[98] Shiva Prasad Kasiviswanathan and Adam Smith. On the 'semantics' of differential privacy: A bayesian formulation. *J. Priv. Confidentiality*, 6(1), 2014.

[99] Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Trans. Database Syst.*, 39(1):3:1–3:36, 2014.

[100] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[101] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In *CRYPTO*, pages 241–257, 2005.

[102] Karl Knopf. *Framework for Differentially Private Data Analysis with Multiple Accuracy Requirements*, page 2890–2892. ACM, 2021.

[103] Solmaz Kolahi. Functional dependency. In *Encyclopedia of Database Systems, Second Edition*. Springer, 2018.

[104] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In *CCS*, pages 1257–1272, 2017.

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

[106] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: A differentially private SQL query engine. *Proc. VLDB Endow.*, 12(11):1371–1384, 2019.

[107] Fragkiskos Koufogiannis, Shuo Han, and George J. Pappas. Gradual release of sensitive data under differential privacy. *CoRR*, abs/1504.00429, 2015.

[108] Fragkiskos Koufogiannis, Shuo Han, and George J. Pappas. Gradual release of sensitive data under differential privacy. *J. Priv. Confidentiality*, 7(2), 2016.

[109] Sanjay Krishnan, Jiannan Wang, Michael J. Franklin, Ken Goldberg, and Tim Kraska. Privateclean: Data cleaning and differential privacy. In *SIGMOD*, pages 937–951. ACM, 2016.

[110] K.E. Lauter, W. Dai, and K. Laine. *Protecting Privacy Through Homomorphic Encryption*. Springer International Publishing AG, 2021.

[111] Jaewoo Lee and Christopher W. Clifton. Top-k frequent itemsets via differentially private fp-trees. In *SIGKDD*, pages 931–940. ACM, 2014.

[112] Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. A data- and workload-aware query answering algorithm for range queries under differential privacy. *Proc. VLDB Endow.*, 7(5):341–352, 2014.

[113] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134. ACM, 2010.

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

[115] Haoran Li, Li Xiong, Lifan Zhang, and Xiaoqian Jiang. Dpsynthesizer: Differentially private data synthesizer for privacy preserving data sharing. *Proc. VLDB Endow.*, 7(13):1677–1680, 2014.

[116] Ninghui Li, Wahbeh H. Qardaji, Dong Su, and Jianneng Cao. Privbasis: Frequent itemset mining with differential privacy. *Proc. VLDB Endow.*, 5(11):1340–1351, 2012.

[117] Katrina Ligett, Seth Neel, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. Accuracy first: Selecting a differential privacy level for accuracy constrained ERM. In *NIPS*, pages 2566–2576, 2017.

[118] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.

[119] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. *IACR Cryptology ePrint Archive*, 2008:197, 2008.

[120] Jixue Liu, Jiuyong Li, Chengfei Liu, and Yongfeng Chen. Discover dependencies from data - A review. *IEEE Trans. Knowl. Data Eng.*, 24(2):251–264, 2012.

[121] Stéphane Lopes, Jean-Marc Petit, and Lotfi Lakhal. Efficient discovery of functional dependencies and armstrong relations. In *EDBT*, 2000.

[122] Ashwin Machanavajjhala, Daniel Kifer, John M. Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, pages 277–286. IEEE Computer Society, 2008.

[123] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Proc. VLDB Endow.*, 11(10):1206–1219, 2018.

[124] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. HDMM: optimizing error of high-dimensional statistical queries under differential privacy. *CoRR*, abs/2106.12118, 2021.

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

[126] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30. ACM, 2009.

[127] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[128] Ilya Mironov. Rényi differential privacy. In *CSF*, pages 263–275. IEEE Computer Society, 2017.

[129] Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism. *CoRR*, abs/1908.10530, 2019.

[130] Dan Murray. *Tableau Your Data! Fast and Easy Visual Analysis with Tableau Software*. Wiley Publishing, 1st edition, 2013.

[131] Moni Naor and Benny Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006.

[132] Noel Novelli and Rosine Cicchetti. FUN: an efficient algorithm for mining functional and embedded dependencies. In *ICDT*, pages 189–203, 2001.

[133] National Institute of Standards and Technology. Differential privacy synthetic data challenge, 2018.

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

[135] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[136] Thorsten Papenbrock, Tanja Bergmann, Moritz Finke, Jakob Zwiener, and Felix Naumann. Data profiling with metanome. *PVLDB*, 2015.

[137] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proc. VLDB Endow.*, 8(10):1082–1093, 2015.

[138] Thorsten Papenbrock and Felix Naumann. A hybrid approach to functional dependency discovery. In *SIGMOD*, 2016.

[139] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*. OpenReview.net, 2017.

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

[141] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *EUROCRYPT*, pages 248–259, 1993.

[142] Eduardo H. M. Pena, Eduardo Cunha de Almeida, and Felix Naumann. Discovery of approximate (and exact) denial constraints. *Proc. VLDB Endow.*, 13(3):266–278, 2019.

[143] NhatHai Phan, Yue Wang, Xintao Wu, and Dejing Dou. Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In *AAAI*, pages 1309–1316. AAAI Press, 2016.

[144] Haoyue Ping, Julia Stoyanovich, and Bill Howe. Datasynthesizer: Privacy-preserving synthetic datasets. In *SSDBM*, pages 42:1–42:5. ACM, 2017.

[145] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis. *Proc. VLDB Endow.*, 7(8):637–648, 2014.

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

[147] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. Relationship privacy: output perturbation for queries with joins. In *PODS*, pages 107–116. ACM, 2009.

[148] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

[149] Jiahui Ren, Xian Xu, Zhihuan Yao, and Huiqun Yu. Recommender systems based on autoencoder and differential privacy. In *COMPSAC*, pages 358–363. IEEE, 2019.

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

[151] Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. A formal framework for probabilistic unclean databases. In *ICDT*, pages 6:1–6:18, 2019.

[152] Krishna Sampigethaya and Radha Poovendran. A survey on mix networks and their secure applications. *Proceedings of the IEEE*, 94(12):2142–2181, 2006.

[153] Hemant Saxena, Lukasz Golab, and Ihab F. Ilyas. Distributed discovery of functional dependencies. In *ICDE*, pages 1590–1593, 2019.

[154] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, pages 2242–2251. IEEE Computer Society, 2017.

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

[156] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[157] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. DP-CGAN: differentially private synthetic data and label generation. *CoRR*, abs/2001.09700, 2020.

[158] Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer series in statistics. Springer, 2009.

[159] Jonathan Ullman and Salil P. Vadhan. Pcps and the hardness of generating private synthetic data. In *TCC*, pages 400–416, 2011.

[160] Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to association rule mining. *J. Comput. Secur.*, 13(4):593–622, 2005.

[161] Elisabet Lobo Vesga, Alejandro Russo, and Marco Gaboardi. A programming language for data privacy with accuracy estimations. *ACM Trans. Program. Lang. Syst.*, 43(2):6:1–6:42, 2021.

[162] Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Zhiwei Steven Wu. New oracle-efficient algorithms for private synthetic data release. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9765–9774. PMLR, 2020.

[163] Christopher Waites. Pyvacy: Towards practical differential privacy for deep learning. *https://github.com/ChrisWaites/pyvacy*, 2019.

[164] Oliver Williams and Frank McSherry. Probabilistic inference and differential privacy. In *NIPS*, pages 2451–2459, 2010.

[165] Richard Wu, Aoqian Zhang, Ihab F. Ilyas, and Theodoros Rekatsinas. Attention-based learning for missing data imputation in holoclean. In *MLSys*, 2020.

[166] Catharine Wyss, Chris Giannella, and Edward L. Robertson. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances - extended abstract. In *DaWaK*, 2001.

[167] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. Knowl. Data Eng.*, 23(8):1200–1214, 2011.

[168] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *CoRR*, abs/1802.06739, 2018.

[169] A. C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167, 1986.

[170] Hong Yao, H. J. Hamilton, and C. J. Butz. Fdmine: discovering functional dependencies in a database using equivalences. In *ICDM*, pages 729–732, 2002.

[171] Sandeep Yaramakala and Dimitris Margaritis. Speculative markov blanket discovery for optimal feature selection. In *ICDM*, pages 809–812, 2005.

[172] Chen Zeng, Jeffrey F. Naughton, and Jin-Yi Cai. On differentially private frequent itemset mining. *Proc. VLDB Endow.*, 6(1):25–36, 2012.

[173] Dan Zhang, Ryan McKenna, Ios Kotsogiannis, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. EKTELO: A framework for defining differentially-private computations. In *SIGMOD*, pages 115–130. ACM, 2018.

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

[175] Xinyang Zhang, Shouling Ji, and Ting Wang. Differentially private releasing via deep generative model. *CoRR*, abs/1801.01594, 2018.

[176] Yunjia Zhang, Zhihan Guo, and Theodoros Rekatsinas. A statistical perspective on discovering functional dependencies in noisy data. In *SIGMOD*, pages 861–876. ACM, 2020.

[177] Tianqing Zhu, Gang Li, Wanlei Zhou, and Philip S. Yu. Differentially private data publishing and analysis: A survey. *IEEE Trans. Knowl. Data Eng.*, 29(8):1619–1638, 2017.