DOI: 10.1561/9781638284772.ch4

Chapter 4

Data Release and Synthetic Data

By Yuchao Tao, David Pujol and Ashwin Machanavajjhala

4.1 Introduction

Organizations and researchers often need to extract valuable insights from sensitive datasets while ensuring the privacy of individuals. As discussed in the previous chapters, Differential Privacy (DP) offers a robust mathematical framework to balance this trade-off between data utility and privacy protection. However, implementing DP in practical scenarios presents significant challenges, especially when dealing with complex queries, high-dimensional data, or large query workloads.

One of the primary challenges is answering as many queries as accurately as possible under a fixed privacy budget. Each query consumes a portion of the privacy budget, and adding noise to each query individually can accumulate excessive noise, degrading the overall utility of the results. For instance, if a statistical database is subjected to numerous queries, even with noise added to each answer, an attacker might reconstruct significant portions of the original data, leading to privacy breaches as highlighted by the fundamental law of information recovery. Consider a scenario where a government agency wants to release statistical information about its citizens, such as average income levels, education statistics, or health metrics. Directly answering each statistical query with added noise may either consume

the entire privacy budget quickly or result in answers that are too noisy to be useful. Moreover, complex queries involving high sensitivity or structured data exacerbate this problem, necessitating advanced mechanisms that can efficiently manage the privacy-accuracy trade-off.

To tackle these challenges, researchers have developed a variety of DP mechanisms tailored to specific settings and tasks. There is no universal mechanism optimal for all scenarios; instead, the choice of mechanism depends on factors such as the nature of the queries, the dimensionality of the data, and whether the queries are processed in an online or offline manner. Additionally, ensuring consistency among query answers and addressing the needs of data analysts (e.g., through synthetic data generation) are crucial considerations in mechanism design. This chapter provides an overview of these challenges and solutions; it reviews some important DP mechanisms designed to balance privacy and utility under various scenarios.

Overview of the Chapter

We begin by motivating the problem and highlighting the inherent challenges in balancing query accuracy with privacy preservation in Section 4.1. Next, in Section 4.2, we examine the factors influencing privacy and utility, such as data dimensionality, query types, and the importance of consistency in query answers. We then delve into workload answering mechanisms (Section 4.3) and data-dependent algorithms that adapt to the underlying data distribution to improve accuracy (Section 4.4). The chapter then proceeds to address online query answering, focusing on mechanisms like Private Multiplicative Weights that handle sequences of adaptively chosen queries while managing the privacy budget effectively (Section 4.5). Finally, we also examine approaches for generating synthetic data under differential privacy (Section 4.6, and review how to answer non-linear queries in Section 4.7, discussing techniques like smooth sensitivity and Lipschitz extensions, which provide ways to accurately answer complex queries while maintaining differential privacy.

4.2 Problem Motivation

The fundamental law of information recovery states [DN03] that a majority of records in a database of size n can be reconstructed when $n\log(n)^2$ queries are answered by the statistical database, even if each answer has up to $O(\sqrt{n})$ error. This either limits the amount of queries that can be answered (even under differential privacy) or the maximum amount of privacy budget that can be used, since increasing either increases the risk of a privacy breach. As such, when answering queries under differential privacy, the fundamental goal of a privacy preserving

mechanism is to answer as many queries as accurately as possible while satisfying differential privacy under a fixed setting of the privacy loss parameter ε.

While the Laplace and Exponential mechanisms are useful primitives for answering individual differentially private queries, they become inefficient when used to answer either queries with high (or unbounded) sensitivity, large sets of queries, structured queries, or queries on structured data. In this chapter, we describe several classes of privacy preserving mechanisms meant to answer queries in different settings. Of these, we will see mechanisms designed to answer large sets of queries by reducing them to smaller sets, mechanisms designed to generate synthetic data which may be queried infinitely many times, mechanisms designed for high dimensional sparse data or particularly dense data, and mechanisms that are designed to answer adaptive queries chosen over time among others. Each of these settings is distinct and poses their own technical problems which come with their own solutions. There is no single differentially private mechanism which is optimal in all settings [Hay+16]; instead there are mechanisms specialized for specific settings and specific tasks which are all designed to maintain privacy under a fixed privacy budget.

4.2.1 Basic Notations and Definitions

We consider a database D as a single tabular data with m attributes and n tuples. The domain for each attribute A_i is Σ_i, and the domain for a tuple t is thus $\Sigma = \Sigma_1 \times \Sigma_2 \ldots \Sigma_m$. A statistical query q takes the database D as input and output a single scalar or a vector of real numbers.

For the majority of this chapter, we focus on linear queries. A linear query can be expressed as $\sum_{t \in D} \phi(t)$, where ϕ is an arbitrary scalar function. When ϕ is in $\{0, 1\}$, this becomes a predicate counting query. When ϕ is a range, this becomes a range counting query.

A linear counting query can also be expressed as a SQL query "SELECT COUNT(*) FROM D WHERE ϕ" to count the number of tuples in D that match the predicate ϕ. A non-linear query has a non-linear transformation of the data before aggregation. For example, consider a table EDGE(from, to) and a self-join counting query "SELECT COUNT(*) FROM EDGE AS E1, EDGE AS E2 WHERE E1.to = E2.from", this query is a non-linear query since self-join is a non-linear transformation.

A workload of queries is a set of queries. When the workload is a set of predicate counting queries with disjoin predicates, this becomes a histogram query. When these disjoint predicates cover the full domain of multiple attributes, this becomes a marginal query over those attributes.

4.3 Different Dimensions of Problems

It is not yet known whether there is a single mechanism that can provide strong utility guarantee for all problems while satisfying differential privacy. These problems could vary in scope and complexity. For example, one problem is to answer a set of statistical queries, and another problem is to release a synthetic data set such that it could be used to answer a set of statistical queries. Meanwhile, these statistical queries could either be linear queries or non-linear queries. From these examples, we observe that problems could be categorized by different characteristics on different dimensions. In this section, we discuss the problems that are commonly studied in differential privacy community in different dimensions.

- **Synthetic data vs query answering.** When we are designing a system that can both protect the data privacy and provide utility for the data analysts, we are facing two choices: one is to make the system interactive, which is to take the queries from the data analysts and return direct query answers. Another is to make the system non-interactive, which is to release a synthetic data set which data analysts can use to answer queries. In the interactive setting, the system can choose a mechanism tailored to the queries of interest [KMHM17]. However, the privacy budget may run out and no future queries could be answered. When synthetic data is released, arbitrary queries or even complex learning tasks can be performed on the synthetic data set. However, the synthetic dataset will not be accurate for all query/analytics tasks.

  There is a large scope of query answering problems that are studied in the differential privacy community, such as range counting queries [Hay+16], marginal queries, graph queries, linear regressions and so on. For the synthetic data generation problem, approaches include density estimation, using domain decomposition, probabilistic graphical models, ML approaches like GANs, and more recently using GPTs/LLMs.
- **Low vs high dimension.** The dimensionality of the data dictates the type of DP algorithm one might use for answering statistical queries. In high dimensional data, the data is quite sparse. Take k-marginal queries on a table with m attributes as example. A k-marginal query is a histogram over k attributes, which lists the counts for all possible domain values for k attributes. If k is 1, this is a histogram over one attribute. When k is getting large, the histogram size is getting exponentially larger, and the count for each bin is almost zero everywhere. Adding Laplace noise to all bin counts to release the k marginal query under DP will result in a relative error that is so high as to make the noisy release useless.

For low dimensional queries such as 1- or 2-dimensional range counting queries, a variety of differentially private mechanisms exists. [Hay+16]. When the query dimension is getting larger, researchers consider a low dimensional representation of the high dimensional queries, such as graphical models [MSM19].

- **Linear vs non-linear queries.** Queries like histogram, marginal and range counting queries are all linear queries. Linear queries are queries that can be expressed as a linear combination of individual weights. The global sensitivity for a linear query is thus bounded by the maximal weight times the number of queries. However, for non-linear queries, the global sensitivity could be much higher or even unbounded. For example, a triangle counting query, which is to count the number of triangles in a graph, has global sensitivity unbounded since there always exists a graph such that adding a node or edge can increase the number of triangles by an arbitrary high number. Adding Laplace noise scaled to the global sensitivity is thus meaningless in this case, so for non-linear queries a different mechanism design should be considered.
- **Online vs offline.** Depending on whether queries arrive all at once or one by one, we separate these two cases as offline mode and online mode, respectively. In the offline mode, all queries are given as the input and the mechanism can take all queries into account to optimize the query results holistically. In the online mode, queries are given as a sequence $q_1, q_2, \ldots, q_n$ that arrives one by one. The total number of queries n may not be known in advance. The query sequence can also be adaptive; i.e., q_i may depend on the answers $a_1, \ldots, a_{i-1}$ for the past queries. Since it is unknown what the future queries are, if most of the privacy budget is spent on answering the early arriving queries accurately, future queries may be answered with poor accuracy. However, if future queries are all similar to the early arriving queries, it is not necessary to save the privacy budget for the early arriving queries. In contrast, in the offline mode, all the queries are known in advance and thus one can use that information to determine an optimal strategy to construct DP answers. Hence, the techniques used to answer queries in the online mode differ from the methods used to answer a batch of queries in the offline mode.
- **Consistency.** When answering multiple queries under DP, the answers released by a differentially private mechanism may not be consistent due to the injected noise; e.g., the noisy count of men and the noisy count of women may not add up to be equal to the noisy count of the total population. Data inconsistency reduces the utility of differential privacy as it causes conflicts in the data analysis. On the other hand, it implies that we have injected unnecessary extra noise to our query answers. Thus, in several settings, there is a need to ensure consistent query answering of multiple queries.

In some cases, enforcing consistency can both improve the utility and reduce the noise, but in some settings it can introduce bias in the released statistics [Puj+20; ZHF21].

4.4 Workload Answering Mechanisms

In this section, we consider answering a workload of linear queries under differential privacy. To simplify the analysis, we use a vector representation of the data and query. Recall that we define a database D as a collection of n tuples, where each tuple is from the domain Σ with N unique domain values. We can also represent a database D as a histogram, where each bin i corresponds to one unique domain value $t_i \in \Sigma$, and its bin count corresponds to how many tuples in D matches to that domain value. We denote this histogram as $x = [x_1, \ldots, x_N]$. The size of this histogram N is the size of the domain Σ. Given this representation of D, we can rewrite the answer of a linear query q from $\sum_{t \in D} \phi(t)$ to $\Sigma_{i=1}^{N} \phi(t_i) x_i$, which is a vector product with x. Therefore, we consider a query as a N dimensional vector. For a workload with d queries, we express it as a $d \times N$ matrix F. The answer for this workload is thus given as Fx.

4.4.1 Lower Bound of Error

Suppose the global sensitivity of the workload F is d. A typical mechanism to answer F on x in an ε-differential private way is to apply d-dimension Laplace mechanism [DMNS06] with sensitivity d. Let error be defined as the expected ℓ_2 distance of the output d-dimensional vectors from the ground truth Fx and the mechanism output. The error for Laplace mechanism in this setting is $d\sqrt{d}/\varepsilon$. A proof sketch is given as follows: Suppose $Z_1, Z_2, \ldots, Z_d$ are i.i.d random variables following the Laplace distribution $Lap(d/\varepsilon)$. The error of Laplace mechanism is thus given as $E\left[\sqrt{\sum_{i=1}^{d} Z_i^2}\right] \leq \sqrt{E\left[\sum_{i=1}^{d} Z_i^2\right]} = \sqrt{d \cdot 2 \cdot (d/\varepsilon)^2} = O(d\sqrt{d}/\varepsilon)$. The first inequality is credit to Jensen inequality.

A lower bound of error in this setting is given by Moritz, Hardt and Kunal Talwar [HT10]. They give a lower bound as $\Omega\left(\min\{d\sqrt{d}/\varepsilon, d\sqrt{\log(N/d)}/\varepsilon\}\right)$ when $d \leq N$. The proof of this lower bound is based on convex geometry. Especially, consider B_1^N as a unit ℓ_1 ball, and K as a convex polytope by $K = FB_1^N$, the lower bound of error is given as $\Omega\left(d\sqrt{d}/\varepsilon \cdot Vol(K)^{1/d}\right)$, where $Vol(K)$ is the volume of K. The idea behind the proof is as follows. Consider each database x as a point in B_1^N, and B_x is a ball centered around Fx with some radius such that the probability that x is mapped into B_x by the mechanism is more than $1/2$. To

satisfy ε-differential privacy, the probability that a zero database 0 is mapped into the same B_x ball should be bounded by a constant factor, and it is held for any x. According to the packing theorem, we can have exponentially many such balls in K that are disjoint, and this number is controlled by the ball radius. Since they are disjoint, the probability that 0 is mapped into the union of these balls is bounded by 1, which means the number of such balls should be bounded. Since the number of such balls is related to the ball radius, this bound gives a lower bound on the ball radius. Intuitively, a larger ball radius indicates lager error for the mechanism, thus bounding the lower bound of the ball radius is associated with bounding the lower bound of the mechanism error.

4.4.2 Select Measure Reconstruct

When answering a workload of queries it is sometimes impractical to answer each individual query directly. Sometimes the workload has a very high sensitivity and thus requires a high amount of noise. Sometimes queries are very similar and share large amounts of data and answering each of these queries would result in a large amount of noise and some inconsistent query answers. In cases like these mechanisms often invoke a design paradigm known as the Select Measure Reconstruct Paradigm. A mechanism which invokes this paradigm first **Selects** an alternative set of queries to answer known as the strategy. This set of queries is usually smaller and having a lower sensitivity than the original workload and as such requires less noise to answer directly. The mechanism then **Measures** these strategy query answers using a differentially private primitive such as the Laplace mechanism. Then the mechanism uses the noisy answers to the strategy workload to **Reconstruct** the original workload. Mechanisms which invoke this paradigm perform well when answering large numbers of related queries.

4.4.3 The Hierarchical Mechanism

One of the first mechanisms which invoked the Select Measure Reconstruct Paradigm is the Hierarchical Mechanism [HRMS09] often called H_b for short. This mechanism was designed to answer large amounts of range queries in an efficient manner. It does so by repeatedly partitioning the domain into a tree with branching factor b, where individual counts in the databases are leaf nodes and parent nodes represent continuous ranges. The mechanism then answers the range queries represented by each individual node in the tree directly. Any additional range queries can be reconstructed using the answers directly in the tree.

In Figure 4.1 we show an example of a Hierarchical tree with branching factor 2 or H_2 for short. In this example, the database in question has domain size of 8. The

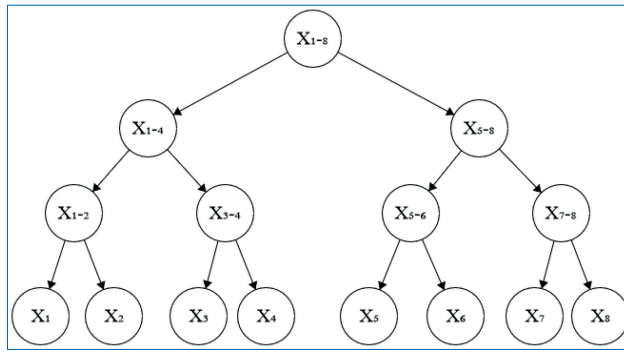

Figure 4.1. An example of an H_2 tree. The root node contains the range query over the entire domain. It's left child node contains the range query over the first half of the domain while it's right node contains the range query over the second half of the domain. The leaf nodes contain queries over single counts.

root tree here represents the range query containing all the counts in the database, the left child represents the range query containing counts 1, 2, 3, and 4 while the right child is the range query containing counts 5,6,7, and 8. The rest of the nodes are denoted similarly. When answering each the queries associated with each node, the sensitivity is the number of nodes along the path from leaf to root, $log_b(k) + 1$, where k is the domain size. Using the H_b strategy any range query can be answered using at most $2\log(k)$ of the nodes. Therefore if we use the Laplace mechanism to measure the nodes each individual node as an expected error of $O\left(\frac{\log(k)^2}{\varepsilon^2}\right)$. Then any range query using at most $2\log(k)$ nodes has an expected error of $O\left(\frac{\log(k)^3}{\varepsilon^2}\right)$ and as such the workload containing all k^2 possible range queries has an expected error of $O\left(\frac{k^2\log(k)^2}{\varepsilon^2}\right)$.

4.4.4 The Matrix Mechanism

The Matrix Mechanism [Li+15] is a more general use of the Select Measure Reconstruct paradigm, which can be used to answer any set of linear counting queries. The matrix mechanism first represents the database as a column vector, denoted x, where each element in the data vector represents the number of individuals that satisfy some property (or combination of properties). Each individual should only be counted in a single element in the data vector in order to bound sensitivity. Once the database is represented in this data vector form, a linear counting query can be represented as a row vector of the same size, and the answer to the query can be computed as the dot product of the query vector and the data vector. A workload of linear counting queries, denoted W is represented as a $k \times n$ matrix where k is the domain size and n is the number of queries. This matrix is simply constructed by

$$\text{SELECT}\ \{A = \text{Optimize}_{MM}(W)$$
$$\text{MEASURE}\ \begin{cases} a = Ax \\ y = a + \text{Lap}(\|A_1\|/\varepsilon) \end{cases}$$
$$\text{RECONSTRUCT}\ \begin{cases} \bar{x} = A^{+}y \\ ans = a + \text{Lap}(\|A_1\|/\varepsilon) \end{cases}$$

Figure 4.2. The SELECT MEASURE RECONSTRUCT paradigm in the matrix mechanism.

the vertical stack of each query. As such answering the entire workload W of queries can be done by the matrix multiplication Wx. Since the data vector requires that each individual be in only one count, the sensitivity of a workload is the maximum L1 column norm of the matrix denoted as $\|W\|_1$. The matrix mechanism takes in a workload W and selects a full rank strategy matrix A to instead answer. It then directly measures A by taking the vector $y = Ax$ and adding independently sampled Laplace noise. It then reconstructs the answers W by taking the pseudo-inverse of $A(\tilde{y} = A^{+}y)$, and then using the resulting noisy data vector to answer the original workload W. An instantiation of the matrix mechanism as Select Measure Reconstruct paradigm is provided in Figure 4.2.

The expected error of the matrix mechanism is as follows:

$$\frac{2}{\varepsilon^2}\|A\|_1^2\|WA^{+}\|_F^2,$$

where $\|\cdot\|_F$ is the Frobenius norm.

The matrix mechanism is a very general mechanism which can be used in many settings and which can represent many other mechanisms. For example consider answering the workload containing all possible range queries with a data vector of size 4 and a privacy budget of $\varepsilon = 1$. If you were to answer the entire workload directly you would get an expected error of 288. Alternatively you could use the hierarchical mechanism, shown as a matrix strategy in Figure 4.3, and use those queries to reconstruct the range queries. Doing so results in an expected error of 201. Originally the task of finding the optimal strategy matrix was proposed as a semi-definite program with rank constraints which could be solved in $O(k^4(n^4 + k^4))$ time. The High Dimensional Matrix Mechanism (HDMM) [MMHM18] is a variant of the matrix mechanism which restricts the possible search space in order to use a gradient decent method to select a strategy matrix. It does this by searching only the simplified p-identity space. This space includes matrices which consist of the the $n \times n$ identity matrix with an additional p rows appended to them, which are

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.3. An example of an H_2 tree with domain equal to 4 represented as a strategy matrix.

then normalized to have a maximum column norm of 1. This reduces the running time of the gradient descent to be a more feasible $O(p * n^2)$.

4.5 Data Dependent Algorithms

Up until now we have considered Data independent Mechanisms whose expected error is independent of the database itself or any properties thereof. Data dependent mechanisms leverage some property of the data to reduce the overall error of the mechanism. This information can be properties of the data which are known ahead of time, such as bounds on specific values, or this data can be inferred during run time such as the distribution of the data. Any information inferred or learned at run-time must be done in a differentially private manner (using some of the privacy budget) to ensure that mechanism as a whole still satisfies differential privacy. Performance of data dependent mechanisms like the name suggests are dependent on how well the mechanism is suited for the specific data. Some mechanisms perform better on sorted or mostly homogeneous data while other mechanisms perform better on data that follows a specific distribution.

4.5.1 Data and Workload Aware Algorithm

The Data- and Workload-Aware Algorithm (DAWA) [LHMW14] is an example of a data adaptive mechanism for computing range and histogram queries. The mechanism has three major steps. First, the mechanism uses a portion of the budget, ε_1, to partition the data into approximately uniform sections; all the counts in each approximately uniform section are added to a bucket. Then it uses a noisy differentially private subroutine to measure the counts in each bucket, in a workload aware manner, similar to the matrix mechanism. Finally, it expands the buckets back

into histogram form by uniformly distributing the counts in each bucket across the domain of the partition. The key point of DAWA is that instead of adding noise to each individual count in the domain, it instead buckets several points in the domain together and adds a single unit of noise to the entire bucket. In the worst case, where each point in the domain is significantly different than it's surrounding points, each point in the domain is given it's own bucket and thus independent noise for each point is required. This is no better than using the Laplace mechanism on each count using only the remainder of the privacy budget ($\varepsilon - \varepsilon_1$). However, in the best case where the data is almost completely homogeneous DAWA could add all of the histogram to a single bucket resulting in only one instance of noise being used across the entire database. Because of this Dawa performs better on databases which have large continuous sections of nearly homogeneous data or on data which is sorted.

4.5.2 PrivTree: Data Adaptive Spatial Decomposition

For a two-dimensional spatial data, e.g. coordinates in a map, a common class of queries is to ask the number of points in a rectangular range. For a single range counting query, to satisfy differential privacy, a typical solution is to add Laplace noise to the count with sensitivity equal to 1. When we have a workload of range counting queries, if we use Laplace noise to answer each query, we have to divide the privacy budget for each query, which results poor accuracy. As shown in Section 4.4 "Workload Answering Mechanisms", one workaround is to partition the domain into small ranges as defined by the range counting queries and their intersections, so that data can be transformed into a vector and the workload of range counting queries can be expressed as a matrix. Then, we can apply techniques introduced in Section 4.4 to release the query answer. However, when we have a fine-grained workload of range counting queries, we also tend to have a fine-grained partition of the domain, which results a sparse vector representation for the data.

An alternative strategy for answering a workload of range counting queries is to first find a good partition of the data so that within each sub-region, points are close to a uniform distribution and the number of them is sufficiently high comparing to the Laplace noise added in each sub-region. For each range counting queries, the count is reported as the sum of noisy counts in the sub-regions it covered and the fractional counts in the sub-regions it partially intersects. Based on these ideas, Jun Zhang, Xiaokui Xiao and Xing Xie propose PrivTree [ZXX16], which decompose the spatial domain by a quad-tree in a differentially private manner. The algorithm starts from the root, which is the entire domain, and then recursively check and decompose the node. For each node that is being checked, it computes the count of points within the region that is represented by the node, decrease the count by a certain amount according to the depth, add a Laplace noise, and then compare

with a threshold. If it is above the threshold, then split the region into halves for each dimension as child nodes, and repeat the sub-routine for each child node.

The quad-tree constructed by PrivTree is totally data-driven. Unlike the traditional private spatial decomposition algorithm, this approach does not specify the limit of max depth, and the privacy loss also does not depend on the max depth of the tree. If data is very dense at a certain sub-region, PrivTree can go deeper for that sub-region without violating differential privacy. Ideally, what PrivTree returns is a spatial decomposition such that dense regions have a dense decomposition and sparse regions have a sparse decomposition. To answer range counting queries, extra Laplace noise is added to each region. For a range counting query on a dense area, it may contain many noises from the sub-regions. However, since it is a dense area, the total counts in this area is sufficiently higher than the total noise, which means the relative error is not low. Other error could come from the bias due to the uniformity assumption within each region.

4.6 Online Query Answering

So far we have assumed the entire workload of queries is known in advance and the mechanism may leverage any structure in the queries to answer them with minimum error. However, in most query answering settings, we imagine an SQL like environment where an analyst asks one query at a time adaptively choosing the next query from the answer to the previous one. This brings us to the online setting. Unlike offline DP mechanisms, where we assume all the queries are known ahead of time, in the online setting the analyst asks one query at a time and the mechanism must answer the queries as they are given before seeing subsequent queries. There are two key challenges in the online setting; budget management and query optimization.

Due to the fundamental law of information recovery we cannot answer an unlimited number of queries, even with noise added each time. Instead we must answer the entire sequence of queries using a limited and fixed privacy budget. Online differentially private mechanisms must spend the privacy budget over time while still maintaining enough budget to answer queries later. Some mechanisms do this by reusing previous query answers or maintaining a synthetic dataset to answer queries from.

Another challenge for online problems is optimizing overall error across queries. In offline query answering, all the queries are known in advance and the mechanism may optimize across the entire workload at once. By contrast, in online query answering, the mechanism only knows the queries that have already been answered, not any that will be asked in the future. Online mechanisms, however, can leverage previously answered queries when asking new ones. A mechanism can avoid

spending budget on a query if it can be reconstructed from previous queries with high accuracy.

4.6.1 Private Multiplicative Weights

Private Multiplicative [HR10] weights is an example of an online query answering mechanism for linear queries. This mechanism can answer an infinite number of linear queries, even after exhausting the allotted privacy budget. The mechanism begins by creating a differentially private normalized histogram from which it can query without spending privacy budget. At initialization the histogram is created to have all values be equal. Whenever a query is asked the mechanism checks (in a differentially private manner) if the current version of the histogram can answer the query accurately. If the histogram can answer accurately the query is answered directly from the maintained histogram. Since the maintained histogram is always updated in a differentially private manner, it itself is differentially private and as such, any queries asked directly from it are differentially private without spending any additional privacy budget. If the answer from the histogram is not accurate enough, the mechanism spends part of its budget asking the query directly from the true dataset using the Laplace mechanism and uses that direct query answer to update the maintained histogram. This way, budget is only spent if the histogram cannot answer the query efficiently, at which point the histogram is updated to more closely resemble the real dataset. This process continues until the mechanism has exhausted it's entire privacy budget. Once the entire privacy budget is spent, the mechanism outputs the maintained differentially private histogram which may be used to answer the remaining queries.

We say that an online query answering mechanism is (α, β, k) adaptively accurate on database x, if for all k adaptively chosen queries $f_1, f_2 \ldots f_k$, with all but β probability $|a_t - \langle f_t, x\rangle| \leq \alpha$. Where $\langle f_t, x\rangle$ is the mechanisms noisy answer to query f_t on database x and a_t is the true answer of that same query. For any set of parameters $k, \varepsilon, \delta, \beta > 0$, Private multiplicative weights is guaranteed to be (α, β, k) adaptively accurate on any database of size n with $\alpha = O\left(\varepsilon^{-1} n^{-1/2} \cdot \log(1/\delta) \log(1/4)(N) \cdot (\log(k) + \log(1/\beta)\right)$. Where $N = |U|$, the size of the data universe. Likewise, since this holds for adaptively chosen queries, it also holds for non-adaptively chosen queries.

4.7 Synthetic Data

A typical scenario for data analysis is that one data analyst submits some queries to the data curator, and data curator returns the query answers to the analyst. To

ensure data is protected by differential privacy, these query answers are perturbed by some differentially private mechanisms. However, the data curator can also choose another strategy to respond to the request. Instead of returning the noisy query answers, the data curator can also generate a synthetic data set that satisfies differential privacy, and then return it to the data analyst. A synthetic dataset is a completely made up dataset with the same schema as the input dataset and which preserves some statistical properties from the input. The data analyst can evaluate queries on the synthetic data set to get query answers, and it satisfies differential privacy due to the post-processing rule (see Theorem 1.7 of Chapter 1.4.2). The utility of a synthetic data release is associated with a specific task or metric. For example, the utility could be associated with the expected ℓ_2 error of a set of linear queries evaluating on the original data and the synthetic data.

Avrim Blum, Katrina Ligett and Aaron Roth show that when considering answering a class of counting queries C using the synthetic data set while satisfying ε-differential privacy, the lower bound of error only depends on the VC-dimension of the query class C and the privacy factor ε [BLR13]. They define (α, δ)-usefulness for a mechanism with respect to the query class C if the max L_1 query error for any query in C is bounded by α with probability $1 - \delta$. Based on the reconstruction proof for "blatant non-privacy" [DN03], they show that given a database with size $\leq VCDIM(C)/2$, for any ε-differentially private mechanism that is (α, δ)-useful for a query class C, we have $\alpha \geq \frac{1}{2\exp(\varepsilon)+1}$. They also propose the Net mechanism [BLR13] that construct a set of candidate data sets, called α-net, such that for any ground truth data set, there always exists a candidate data set that can accurately answers any query from a fixed query class compared to the ground truth one with L1 error less than α. The Net mechanism then chooses a data set from the α-net using exponential mechanism with score function as the inverse max L1 error for all queries, and thus satisfies differential privacy. However, the size of α-net is often large. The bound given in [BLR13] shows that for any counting class C and any data domain X, the size of minimal α-net is at most $|X|^{O(VCDIM(C)log(1/\alpha)/\alpha^2)}$. It is also computationally infeasible to sample a data set from the α-net according to the exponential mechanism.

Another approach for private synthetic data generation uses generative models, which includes probabilistic graphical models and deep generative models. Approaches based on probabilistic graphical models include Bayesian networks [Zha+17] and Markov networks [CXZX15; Ber+17; MSM19; ZKKW20]. Most of these approaches first learn the probabilistic graphical model structure of the data, and then learn the parameters for the model. Approaches based on deep generative models include DP-AuGM [Che+18], DP-VaeGM [Che+18], DPGAN [Xie+18], and PATE-GAN [JYV18]. These approaches are based on training (variational) autoencoders or generative adversarial networks in a differentially private

way. Other approaches includes estimating the data distribution from the noisy DP query answers, such as MWEM [HLM10] and PGM estimation [MSM19].

4.7.1 PrivBayes

PrivBayes [Zha+17] is one of several methods for generating differentially private synthetic data. It does so through three major steps. First, it splits the entire privacy budget into two separate budgets, ε_1 and ε_2. The first of these budgets is used to learn a Bayesian network capturing the important attribute correlations in the input database in a differentially private manner. The second privacy budget is used to construct noisy versions of the conditional distributions contained in the Bayesian network. From these two steps we are given a differentially private Bayesian network as well as differentially private conditional distribution for each node in the network. In combination the network and the noisy conditional distributions are then used to approximate the distribution of tuples in the original database. Sampling from this distribution creates differentially private synthetic data with a distribution approximately equivalent to the tuples in the original database. Since the first two steps satisfy ε_1 and ε_2 differential privacy respectively and since the third step is a post processing step PrivBayes algorithm as a whole satisfies $\varepsilon = \varepsilon_1 + \varepsilon_2$ differential privacy.

4.7.2 Markov network

Suppose the domain for a tuple is $\mathcal{X}$, we could express this data as a $|\mathcal{X}|$-dimensional full-domain vector, where each cell in the vector indicates the fraction of rows in the data matching a specific tuple value. Assuming N is public and fixed, this $|\mathcal{X}|$-dimensional vector is equivalent to a contingency table of the data, which lists the number of rows matched for all possible values of n attributes.

To generate a synthetic data set under differential privacy, one approach is to perturb the $|\mathcal{X}|$-dimensional full-domain vector using Laplace mechanism, post-process the noisy vector to be non-negative, normalize it to be a probability distribution and then sample a synthetic data set from it. Roughly speaking, the expected statistical distance between the sanitized distribution and the ground truth grows with the ratio $|\mathcal{X}|/N$, where $|\mathcal{X}|$ is the size of the full-domain vector and N is the data size. When $|\mathcal{X}|$ is much larger than the data size N, most cells in the full-domain vector are zero and few cells have a low non-zero counts, thus adding Laplace noise to all cells results large amount of relative error, which further makes the sanitized distribution useless.

An alternative approach is to learn several lower dimensional marginal distributions of the data using differentially private mechanisms, and then infer the full-domain distribution with some principles. Since the full-domain distribution

is of high dimension, it is important to find a computationally tractable expression of the distribution. One such principle to infer the full-domain distribution is to find a distribution such that its marginal distributions are close to the sanitized ones generated by the differentially private mechanisms. Based on these ideas, Ryan McKenna, Daniel Sheldon and Gerome Miklau propose a inference principle based on the undirected graphical model of data and the space of marginal polytope [MSM19]. An undirected graphical model, or Markov network, factorizes the data distribution into a product of factor functions over the cliques of a graph with a normalization. In [MSM19], they assume all the attributes from the noisy marginal distributions given as the input for the full-domain distribution inference are the cliques of the graphical model. The goal is to find the parameters in the factor functions such that the loss of marginal distributions is minimized while the entropy of the full-domain distribution is maximized.

To achieve this goal, the first step is to find a valid set of marginals such that the loss between the found marginals and the noisy marginals are minimized, and then the factor parameters can be derived from the marginals using the maximum entropy rule. A set of marginals is valid if there exists a full-domain distribution such that its marginals match the set of marginals. The space of such valid marginals form a marginal polytope. The optimization problem is thus to find a valid set of marginals with minimum loss on this marginal polytope. The loss is defined as the negative log-likelihood in [MSM19], where the likelihood is about how likely the differentially private mechanism generates the noisy result given a specific marginal. In [MSM19], the optimization problem is solved by an algorithm based on entropic mirror descent algorithm, and further improved by Nesterov's accelerated dual averaging approach. This algorithm also generates the factor parameters at the same time. To generate a synthetic data set, one can sample from the graphical model using the inferred factor parameters.

Learning a Markov network from the noisy marginals not only provides a tractable expression of the full-domain distribution, but also ensure consistency and some accuracy for the further inference of the distribution. However, it is unknown whether a data distribution can always be factorized using a Markov network, and it is also unknown whether the maximum entropy rule is the best option to infer a Markov network given a set of marginal distributions.

4.7.3 Multiplicative Weights Exponential Mechanism

Suppose we are given some initial guess to the full-domain distribution, such as a uniform distribution, and we are also given a query result of q, which maps each tuple in the data to $[-1, +1]$ and sums up, how should we update our initial guess of the full-domain distribution according to what we have observed? One approach is to update the weight of each value in the domain using the multiplicative weight

rule. Section 4.6.1 introduces this approach in on online setting, where queries are coming one by one and the distribution is updated every time if a query answered using the Laplace mechanism on the ground truth data is very different from being answered on the synthetic data set. The query answer that is released is always based on the latest synthetic data set.

Maintaining a synthetic data set or distribution in this way is also an approach to release a private synthetic data set. Here we consider an offline setting as discussed in MWEM [HLM10]. If all the queries are given at once instead of arriving one by one, we can cherry-pick the query which has the maximum error to update the distribution. This query selection is done through the Exponential mechanism, where the score function is the L1 distance between the query answer on the current distribution and the ground truth. Once a query is selected, the query is answered based on the ground truth data using the Laplace mechanism, and the distribution is updated using the multiplicative rule. The algorithm will then repeat again to select a new query and make a new update. If there are in total T such updates, then the privacy budget is divided into $2T$ pieces, one for the EM mechanism and one for the Laplace mechanism in each update. The final distribution or the synthetic data set is the average of all past distribution in the updates. Suppose the domain size of data is $|D|$, the pool size of queries is $|Q|$, the data size is n and the total privacy budget is ε, then there exists a T such that with probability at least $1 - 1/ploy(|Q|)$, the error of any query answered by the final synthetic data set is $O\left(n^{2/3}\left(\frac{\log|D|\log|Q|}{\varepsilon}\right)\right)$

Since the update of distribution is a pure post-processing step, one can apply the multiplicative weight update multiple times using the existing sanitized query answers. The initial guess of the distribution can also be replaced by some public knowledge or another private mechanism. The final distribution can also be replaced by the final updated distribution instead of the averaged one. These variants of the algorithm may improve the performance of MWEM in a practical sense.

4.8 Non-linear query Answering

Non-linear queries are queries that cannot be expressed as the weighted summation of data elements. Examples include quantile queries, join-count queries, and graph queries. Most DP mechanisms are designed for linear queries. For non-linear queries, the global sensitivity is not clear and naively applying Laplace mechanism may fail to follow DP.

4.8.1 Smooth Sensitivity

For a linear query, the global sensitivity is determined by the max weight in the query. For example, the global sensitivity is 1 for a common predicate-counting query where each weight is one or zero. To answer such a linear query under ε-differential privacy, one approach is apply the Laplace mechanism, which adds a Laplace noise scaled to the global sensitivity of the query divided by the privacy budget ε. However, for a non-linear query, the global sensitivity is often much larger. For example, for a median query on a data set with domain $[0, \Lambda]$, the global sensitivity is Λ. Consider a data set with $2n+1$ elements such as $\{0, 0, \ldots, \Lambda, \Lambda\}$ where n are 0 and $n+1$ are Λ. The median is this case is Λ. However, under bounded differential privacy, consider a neighboring database by changing the $n+1^{st}$ element from Λ to 0; the median drops to 0. This example shows that the global sensitivity of median is Λ, and thus releasing the median using Laplace mechanism could result in meaningless outputs when Λ is very large or the number of elements is small. This is surprising, since the median is less sensitive especially for large datasets, but the relative noise introduced by Laplace mechanism is not decreasing as we increase the data size.

One way to reduce the noise needed to achieve differential privacy is to replace global sensitivity with a measure of sensitivity on the specific database instance. Local sensitivity is the maximum query difference between the input database and all its neighboring data sets. For measures like the median, the local sensitivity varies depending on the input database. For the pathological example above, the local sensitivity is still Λ. However, for a database with $2n+1$ elements with the same value in the positions n, $n+1$ and $n+2$, then the local sensitivity is 0! However, the local sensitivity itself is a sensitive information of the data (e.g. for the case of median you can tell the values in positions n, $n+1$ and $n+2$ are the same when local sensitivity is 0). Hence, adding Laplace noise scaled to the local sensitivity is does not guarantee differentially privacy. To connect local sensitivity to differential privacy, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith propose the smooth sensitivity mechanism [NRS07]. Smooth sensitivity is a tight smooth upper bound of the local sensitivity such that itself is e^{β}-Lipschitz for some β: i.e., smooth sensitivity $SS(\cdot)$ has the following property: $SS(D) \geq LS(D)$ and $|SS(D) - SS(D')| \leq e^{\beta} \cdot d(D, D')$ for any data sets D and D', where $LS(D)$ is the local sensitivity of D and $d(D, D')$ is the hamming distance between D and D'. Calibrating noise to smooth sensitivity and adding it to the raw query result achieves (ε, δ)-differential privacy.

Deriving the smooth sensitivity S_f from the local sensitivity LS_f for the query f on some data set is not trivial. Taking the median as an example. Consider a data set with n elements and n is odd. Suppose data is ordered as $\{x_1, x_2, \ldots, x_n\}$. Denote $m = \frac{n+1}{2}$ as the index for the median. The local sensitivity LS_{med} is thus equal to

$\max(x_m - x_{m-1}, x_{m+1} - x_m)$. To derive the smooth sensitivity, [NRS07] considers an extension of local sensitivity as local sensitivity at distance k, $LS_f^{(k)}$, which is the max local sensitivity for the data set that differs by at most k rows with the ground truth data set. The smooth sensitivity of f is thus equal to the max of $e^{-k\beta} LS_f^{(k)}$ for $k \in \{0, 1, \ldots, n\}$. For the median case, its local sensitivity at distance k is given as $LS_{med}^{(k)} = \max_{0 \le t \le k+1} (x_{m+t} - x_{m+t+k-1})$, which can be further used to derive the smooth sensitivity, and the running time for computing its smooth sensitivity is thus $O(n^2)$.

It is not always clear how to efficiently compute the smooth sensitivity. A naive algorithm to find local sensitivity at distance k is to try all possible data set by changing up to k tuples, and for each data set iterate over all neighboring data sets of that data set to compute the local sensitivity. To deploy smooth sensitivity in a real application, it is important to find a computationally efficient algorithm to compute the smooth sensitivity or its approximate upper bound.

4.8.2 Lipschitz Extension

One representative class of non-linear queries is sub-graph counting queries. For example, given an un-directed graph $G(V, E)$, a triangle counting query counts number of triangles in the graph. Other sub-graph counting queries include counting the number of edges, k-stars, and so on. If we think about each node in the graph as the basic element, sub-graph counting queries cannot be expressed as a linear combination of nodes since each node may be involved in multiple sub-graphs.

To answer these queries under differential privacy, a basic solution is to apply Laplace mechanism, which adds the Laplace noise scaled to the global sensitivity. Recall that the definition of global sensitivity for a function f is $\max |f(G) - f(G')|$ for all G and G' that are neighbors. Here we consider G and G' are neighbors if they differ by a single node. This is so called node-DP. Under node-DP, for a graph query like counting the number of triangles, the global sensitivity is unbounded. Even if we assume the total number of nodes is n, the global sensitivity is $\binom{n-1}{2}$ due to a pair of neighboring graphs that are n-clique and $(n-1)$-clique.

If we assume the graph degree is bounded by D and denote this space as $\mathcal{G}^D$, then the global sensitivity of triangle counting query is $\binom{D}{2}$, which is much smaller than $\binom{n-1}{2}$ since $D \ll n$ is common. Usually D can be selected by some common knowledge or some public information, so that it is close to the real degree bound. However, we cannot guarantee this assumption is always true. When this assumption does not hold, we still want our query to be have low sensitivity like when the assumption is true. This leads to the solution based on Lipschitz extension, which is to find a new function that gives the same answer in the original scope and also has

the same global sensitivity[i]. For example, suppose f is defined on graphs from G^D, we can find a new function $\hat{f}$ that takes any graph as input such that $\hat{f}(G) = f(G)$ on $G \in \mathcal{G}^D$ and the global sensitivity $GS(\hat{f}) = GS(f)$.

Finding the Lipschitz extension of graph queries is not trivial. Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova and Adam Smith consider using max flow of a constructed flow graph for answering the number of edges and using linear programming for answering the number of sub-graphs [KNRS13]. Given a graph $G(V, E)$, a flow graph is constructed as a source s, a sink t and two copies of V as V_l and V_r such that the source s is connected to each v_l in V_l with capacity D, each v_r in V_r is connected to the sink t with capacity D and v_l is connected to v_r with unit capacity if $(v_l, v_r) \in E$. The half of the max flow in this graph is a Lipschitz extension of the number of edges for graphs from $\mathcal{G}^D$. For answering the number of a specific sub-graph in G, a Lipschitz extension based on linear programming is considered as follows: Suppose Δ_D is the global sensitivity of answering the number of sub-graph for graphs from $\mathcal{G}^D$. Create variables $x_c \in [0, 1]$ for each possible sub-graph c in G (no matter whether it exists or not). For each node in G, the sum of x_c that is adjacent to that node is bounded by Δ_D. The objective is to maximize the sum of x_c. The optimal value by solving this linear programming is a Lipschitz extension of the sub-graph counting for graphs from $\mathcal{G}^D$.

These examples of Lipschitz extension shows the applicability of Lipschitz extension to graph queries with scalar outputs. For graph queries with multi-dimensional outputs, such as degree distribution, another flow graph based solution is considered [RS16]. Other studies based on Lipschitz extension include [BBDS13; DLL16; DZBJ18]. Studies that involves the similar idea of Lipschitz extension includes [CZ13; Zha+15; Kot+19; THMR20]

4.8.3 Answering SQL Queries

SQL queries are widely existing in the real world data analysis tasks. They provide a rich model of query structures that is far beyond simple linear queries. Privately answering SQL queries is thus challenging. On the other hand, SQL queries are often asked on a relational database with multiple relations. Relations are usually associated with each other by some foreign-primary keys. In a multi-relational database, if we remove one tuple in a relation, it is nature to consider cascade delete as removing tuples in other relations that have foreign keys associated with the tuple. In this case, even for a simple linear query, the privacy analysis is not clear since removing a tuple may cause cascade deletions and the sensitivity for the linear query thus is not always a constant.

i. If the new global sensitivity is s times the original one, it is called Lipschitz extension with stretch s

Kotsogiannis et al. propose a system PrivateSQL for answering SQL queries privately [Kot+19]. It extends differential privacy to multi-relational databases with foreign-primary key constraints, which captures Edge- or Node-DP for graphs [KRSY11; KNRS13]. It also supports complex SQL queries that include JOINS, GROUPBY and correlated sub-queries. To boost accuracy, PrivateSQL considers decomposing queries into multiple workloads. Within each workload, queries are transformed as a set of linear queries on a complex view. To ensure differential privacy on a multi-relational database, PrivateSQL applies the query rewriting technique to the view so that it is equivalent to the standard differential privacy on a single data set and the stability of the view is also bounded. View stability is the maximum change of rows in the view after adding or removing a single private tuple. Since it is well studied about answering a workload of linear queries on a view with stability 1 [Hay+16; MMHM18], it is natural to extend this problem to a view with a higher stability.

Tracking the stability of a view is not trivial, since a view is equivalent to a tree of SQL operations with arbitrary size. Flex gives a rule-based stability calculation strategy by recursively update the stability of a node in the SQL query tree [JNS18]. To capture the stability update due the JOIN operator, Flex also tracks the frequency of attribute values. Since attribute frequency is a sensitive information, the stability of the view cannot be revealed. Therefore, Flex uses smooth sensitivity [NRS07] to perturb the query answer. PrivateSQL extends this rule-based stability calculator with new rules on the JOIN stability update. Furthermore, PrivateSQL computes the attribute frequency in a differentially private approach through sparse vector technique [DR+14] and enforces the frequency bound by injecting truncation operators into the SQL query tree, which allows the view stability to be publicly accessible. An alternative to find a good bounding frequency is through recursive mechanism [CZ13], which is based on a list of max frequencies related to the subsets of the primary private relation with size ranging from zero to full. Another private SQL engine is proposed by Wilson et al. [Wil+19], which considers aggregations other than counting, such as average and quantile. They also consider bounding the user contribution by a fixed number in a join query using reservoir sampling.

4.9 Concluding Remarks

In this chapter, we have elaborated the challenge of releasing statistics and synthetic data using differential privacy, and we have also discussed the multiple dimensions of the problem which includes synthetic vs query answering, low vs high dimension, online vs offline and consistency issues. We further introduce the algorithm design

primitives for different tasks, which includes data dependent algorithms, online query answering algorithms, synthetic data generation algorithms and non-linear query answering algorithms.

References

[BBDS13] J. Blocki, A. Blum, A. Datta, and O. Sheffet. "Differentially private data analysis of social networks via restricted sensitivity". In: Proceedings of the 4th conference on Innovations in Theoretical Computer Science. 2013, pp. 87–96 (cit. on p. 176).

[Ber+17] G. Bernstein, R. McKenna, T. Sun, D. Sheldon, M. Hay, and G. Miklau. "Differentially private learning of undirected graphical models using collective graphical models". In: International Conference on Machine Learning. PMLR. 2017, pp. 478–487 (cit. on p. 170).

[BLR13] A. Blum, K. Ligett, and A. Roth. "A learning theory approach to noninteractive database privacy". In: Journal of the ACM (JACM) 60.2 (2013), pp. 1–25 (cit. on p. 170).

[Che+18] Q. Chen, C. Xiang, M. Xue, B. Li, N. Borisov, D. Kaarfar, and H. Zhu. "Differentially private data generative models". In: arXiv preprint arXiv:1812.02274 (2018) (cit. on p. 170).

[CXZX15] R. Chen, Q. Xiao, Y. Zhang, and J. Xu. "Differentially private high-dimensional data publication via sampling-based inference". In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. 2015, pp. 129–138 (cit. on p. 170).

[CZ13] S. Chen and S. Zhou. "Recursive mechanism: towards node differential privacy and unrestricted joins". In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. 2013, pp. 653–664 (cit. on pp. 176, 177).

[DLL16] W.-Y. Day, N. Li, and M. Lyu. "Publishing graph degree distribution with node differential privacy". In: Proceedings of the 2016 International Conference on Management of Data. 2016, pp. 123–138 (cit. on p. 176).

[DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. "Calibrating noise to sensitivity in private data analysis". In: Theory of cryptography conference. Springer. 2006, pp. 265–284 (cit. on p. 162).

[DN03] I. Dinur and K. Nissim. "Revealing information while preserving privacy". In: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. 2003, pp. 202–210 (cit. on pp. 158, 170).

[DR+14] C. Dwork, A. Roth, et al. "The algorithmic foundations of differential privacy." In: Foundations and Trends in Theoretical Computer Science 9.3-4 (2014), pp. 211–407 (cit. on p. 177).

[DZBJ18] X. Ding, X. Zhang, Z. Bao, and H. Jin. "Privacy-preserving triangle counting in large graphs". In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 2018, pp. 1283–1292 (cit. on p. 176).

[Hay+16] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang. "Principled evaluation of differentially private algorithms using dpbench". In: Proceedings of the 2016 International Conference on Management of Data. 2016, pp. 139–154 (cit. on pp. 159–161, 177).

[HLM10] M. Hardt, K. Ligett, and F. McSherry. "A simple and practical algorithm for differentially private data release". In: arXiv preprint arXiv:1012.4763 (2010) (cit. on pp. 171, 173).

[HR10] M. Hardt and G. N. Rothblum. "A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis". In: 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23–26, 2010, Las Vegas, Nevada, USA. IEEE Computer Society, 2010, pp. 61–70. URL: https://doi.org/10.1109/FOCS.2010.85 (cit. on p. 169).

[HRMS09] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. "Boosting the Accuracy of Differentially-Private Queries Through Consistency". In: CoRR abs/0904.0942 (2009). arXiv: 0904.0942. URL: http://arxiv.org/abs/0904.0942 (cit. on p. 163).

[HT10] M. Hardt and K. Talwar. "On the geometry of differential privacy". In: Proceedings of the forty-second ACM symposium on Theory of computing. 2010, pp. 705–714 (cit. on p. 162).

[JNS18] N. Johnson, J. P. Near, and D. Song. "Towards practical differential privacy for SQL queries". In: Proceedings of the VLDB Endowment 11.5 (2018), pp. 526–539 (cit. on p. 177).

[JYV18] J. Jordon, J. Yoon, and M. Van Der Schaar. "PATE-GAN: Generating synthetic data with differential privacy guarantees". In: International Conference on Learning Representations. 2018 (cit. on p. 170).

[KMHM17] I. Kotsogiannis, A. Machanavajjhala, M. Hay, and G. Miklau. "Pythia: Data dependent differentially private algorithm selection". In: Proceedings of the 2017 ACM International Conference on Management of Data. 2017, pp. 1323–1337 (cit. on p. 160).

[KNRS13] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. "Analyzing graphs with node differential privacy". In: Theory of Cryptography Conference. Springer. 2013, pp. 457–476 (cit. on pp. 176, 177).

[Kot+19] I. Kotsogiannis, Y. Tao, X. He, M. Fanaeepour, A. Machanavajjhala, M. Hay, and G. Miklau. "Privatesql: a differentially private sql query engine". In: Proceedings of the VLDB Endowment 12.11 (2019), pp. 1371–1384 (cit. on pp. 176, 177).

[KRSY11] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev. "Private analysis of graph structure". In: Proceedings of the VLDB Endowment 4.11 (2011), pp. 1146–1157 (cit. on p. 177).

[LHMW14] C. Li, M. Hay, G. Miklau, and Y. Wang. "A Data- and Workload-Aware Algorithm for Range Queries Under Differential Privacy". In: CoRR abs/1410.0265 (2014). arXiv: 1410.0265. URL: http://arxiv.org/abs/1410.0265 (cit. on p. 166).

[Li+15] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi. "The matrix mechanism: optimizing linear counting queries under differential privacy". In: VLDB J. 24.6 (2015), pp. 757–781. URL: https://doi.org/10.1007/s00778-015-0398-x (cit. on p. 164).

[MMHM18] R. McKenna, G. Miklau, M. Hay, and A. Machanavajjhala. "Optimizing Error of High-dimensional Statistical Queries Under Differential Privacy". In: PVLDB 11.10 (2018) (cit. on pp. 165, 177).

[MSM19] R. McKenna, D. Sheldon, and G. Miklau. "Graphical-model based estimation and inference for differential privacy". In: International Conference on Machine Learning. PMLR. 2019, pp. 4435–4444 (cit. on pp. 161, 170–172).

[NRS07] K. Nissim, S. Raskhodnikova, and A. Smith. "Smooth sensitivity and sampling in private data analysis". In: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing. 2007, pp. 75–84 (cit. on pp. 174, 175, 177).

[Puj+20] D. Pujol, R. McKenna, S. Kuppam, M. Hay, A. Machanavajjhala, and G. Miklau. "Fair decision making using privacy-protected data". In: FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020. Ed. by M. Hildebrandt, C. Castillo, E. Celis, S. Ruggieri, L. Taylor, and G. Zanfir-Fortuna. ACM, 2020, pp. 189–199. URL: https://doi.org/10.1145/3351095.3372872 (cit. on p. 162).

[RS16] S. Raskhodnikova and A. Smith. "Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism". In: 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS). IEEE. 2016, pp. 495–504 (cit. on p. 176).

[THMR20] Y. Tao, X. He, A. Machanavajjhala, and S. Roy. "Computing Local Sensitivities of Counting Queries with Joins". In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020, pp. 479–494 (cit. on p. 176).

[Wil+19] R. J. Wilson, C. Y. Zhang, W. Lam, D. Desfontaines, D. Simmons-Marengo, and B. Gipson. "Differentially private sql with bounded user contribution". In: arXiv preprint arXiv:1909.01917 (2019) (cit. on p. 177).

[Xie+18] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou. "Differentially private generative adversarial network". In: arXiv preprint arXiv:1802.06739 (2018) (cit. on p. 170).

[Zha+15] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. "Private release of graph statistics using ladder functions". In: Proceedings of the 2015 ACM SIGMOD international conference on management of data. 2015, pp. 731–745 (cit. on p. 176).

[Zha+17] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. "Privbayes: Private data release via bayesian networks". In: ACM Transactions on Database Systems (TODS) 42.4 (2017), pp. 1–41 (cit. on pp. 170, 171).

[ZHF21] K. Zhu, P. V. Hentenryck, and F. Fioretto. "Bias and Variance of Post-processing in Differential Privacy". In: AAAI Conference on Artificial Intelligence. AAAI Press, 2021, pp. 11177–11184. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17333 (cit. on p. 162).

[ZKKW20] H. Zhang, G. Kamath, J. Kulkarni, and S. Wu. "Privately learning Markov random fields". In: International Conference on Machine Learning. PMLR. 2020, pp. 11129–11140 (cit. on p. 170).

[ZXX16] J. Zhang, X. Xiao, and X. Xie. "Privtree: A differentially private algorithm for hierarchical decompositions". In: Proceedings of the 2016 International Conference on Management of Data. 2016, pp. 155–170 (cit. on p. 167).