New Algorithms for Preserving Differential Privacy

Aaron Roth

CMU-CS-10-135

July 2010

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Avrim Blum, Chair
Cynthia Dwork, Microsoft Research SVC
Anupam Gupta
Larry Wasserman

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This thesis has been supported in part by an NSF Graduate Research Fellowship

Keywords: Differential Privacy, Algorithms, Game Theory

Abstract

In this thesis, we consider the problem of how one should perform computations on *private* data. We specifically consider algorithms which preserve the recent formalization of privacy known as *differential privacy*. The fundamental tradeoff that we consider is that of *privacy* and *utility*. For which tasks can we perform useful computations while still preserving privacy, and what exactly is the tradeoff between usefulness and privacy? We study this tradeoff for statistical query release, both offline and online, as well as for many standard combinatorial optimization tasks.

Acknowledgements

My career as a computer scientist began in college. I didn't know what I wanted to study, but I was extremely fortunate to be at Columbia, where Rocco Servedio was on the faculty; by the time I had graduated, I had taken all of his courses, and was convinced that I wanted to be a computer scientist. When it came time to decide where to go to graduate school, Rocco assured me that I could not do better than to go to Carnegie Mellon and to work with Avrim Blum. He was of course right.

I could not have asked for a better advisor than Avrim. He is an excellent teacher, and has an amazing intuition for how to correctly pose problems, such that their solutions seem elegant and inevitable in retrospect. He was an invaluable source of inspiration and encouragement throughout my time at Carnegie Mellon, showed me how to find and solve problems, and is a role model as teacher and researcher.

I am thankful for the collaborators that I have had the pleasure of working with throughout my career as a graduate student, including Moshe Babaioff, Nina Balcan, Liad Blumrosen, Anupam Gupta, Christine Chung, MohammadTaghi Hajiaghayi, Adam Kalai, Katrina Ligett, Yishay Mansour, Frank McSherry, Kirk Pruhs, Tim Roughgarden, Grant Schoenebeck, and Kunal Talwar. Moshe Babaioff, Liad Blumrosen, and Adam Kalai hosted me for terrific internships at Microsoft Research, and working with them has been a pleasure. They have been a terrific source of advice, both in and outside of research. Tim Roughgarden hosted me for a visit at Stanford for what was my most productive three weeks yet: the result was Chapter 4 of this thesis. Tim is an amazing researcher, and a joy to work with. While at Microsoft, I also had the pleasure of collaborating with Frank McSherry and Kunal Talwar, who both combine a measure of technical skill and personal friendliness that makes working with them a real joy. Anupam Gupta is incredibly generous with his time, insight, talent, and advice: he is an ideal collaborator, and has helped me enormously throughout my time in graduate school. Much of my work in graduate school was in collaboration with my fellow graduate student, Katrina Ligett. In my first year in graduate school, at the first conference that I attended, she introduced me to researchers when I didn't know anybody. She has been an ideal collaborator, and graduate school would not have been the same without our many hours of research discussions. And of course, thank you to all of the members of my thesis committee: Avrim, Cynthia, Anupam, and Larry.

My friends and family have been a constant source of support and encouragement. My parents have supported and inspired me throughout my life – I can only hope to one day achieve a small measure of their accomplishments. Most of all, I thank Cathy for supporting me and being there throughout college, graduate school, and beyond. You

make everything worthwhile.

Finally, a personal thank you to anyone reading this thesis.

Contents

1 Introduction **1**

1.1 Overview of the Thesis 2

1.1.1 Releasing Statistical Information Privately 2

1.1.2 Performing Combinatorial Optimization Privately 3

2 Background, Definitions, and Related Work **5**

2.1 Privacy: Why and How 5

2.2 Differential Privacy 8

2.3 Predicate Queries 12

2.4 Related Work in Differential Privacy 14

3 Answering Exponentially Many Predicate Queries **17**

3.1 Introduction 17

3.1.1 Related Work 19

3.1.2 Motivation from Learning Theory 20

3.1.3 Organization 20

3.2 Definitions 21

3.3 General release mechanism 22

3.4 Interval queries 24

3.5 Lower bounds 27

3.6 Answering Halfspace Queries 28

3.7 Distributional Privacy 31
3.7.1 Relationship Between Definitions 32
3.8 Lower Bounds 34
3.8.1 VC Dimension Lower Bounds 34
3.8.2 Parity: a $1/\sqrt{n}$ lower bound 34
3.9 Conclusions 37

4 Answering Queries Interactively: The Median Mechanism 39
4.1 Introduction 39
4.1.1 Our Results 41
4.1.2 The Main Ideas 41
4.1.3 Related Work 42
4.2 The Median Mechanism: Basic Implementation 44
4.3 Analysis of Median Mechanism 46
4.3.1 Utility of the Median Mechanism 46
4.3.2 Privacy of the Median Mechanism 49
4.4 The Median Mechanism: Polynomial Time Implementation 54
4.4.1 Redefining the sets C_i 55
4.4.2 Usefulness for Almost All Distributions 57
4.4.3 Usefulness for Finite Databases 59
4.5 Conclusion 62

5 Differential Privacy and the Fat-Shattering Dimension of Linear Queries 63
5.1 Introduction 63
5.1.1 Related Work and Our Results 64
5.2 Preliminaries 65
5.2.1 Fat Shattering Dimension 66
5.3 Lower Bound 67
5.4 Upper Bound 70

5.5 Conclusion 75

6 Data Release in the Local Privacy Model **77**

6.1 Introduction 77

6.2 The Local Privacy Model 78

6.3 Conclusion: The Local Privacy Barrier for Efficient Release Mechanisms 83

7 Differentially Private Combinatorial Optimization **85**

7.1 Introduction 85

7.1.1 Our Results 86

7.1.2 Related Work 90

7.2 Definitions 90

7.2.1 The Exponential Mechanism 90

7.3 Private Min-Cut 91

7.3.1 Lower Bounds 93

7.4 Private k-Median 93

7.4.1 k-Median Lower Bound 95

7.5 Vertex Cover 96

7.5.1 Approximating Solutions: Unweighted Vertex Cover 97

7.5.2 Vertex Cover Lower Bounds 100

7.5.3 Weighted Vertex Cover 101

7.6 Unweighted Vertex Cover Algorithm: An Alternate View 105

7.7 Set Cover 106

7.7.1 Unweighted Set Cover 107

7.7.2 Weighted Set Cover 109

7.7.3 Removing the Dependence on W 113

7.7.4 Lower bounds 114

7.7.5 An Inefficient Algorithm for Weighted Set Cover 115

7.8 Combinatorial Public Projects (Submodular Maximization) 117

7.8.1 Utility Analysis 118
7.8.2 Privacy Analysis 119
7.8.3 Truthfulness 121
7.8.4 Lower Bounds 121

Chapter 1

Introduction

Consider the problem faced by the administrator of a database consisting of sensitive personal information – say a collection of patient medical records. The data may be extremely valuable: perhaps it can be used to learn epidemiological information that could be used to save lives, or to optimize the distribution of heterogeneous resources across hospitals that serve different patient populations. However, the database administrator may not be able to make use of the vast literature on learning and optimization, because using such algorithms may reveal information about the patients in the data set. Revealing their private data might be unacceptable from a moral, legal, and financial standpoint.

What then is the hospital administrator to do? In order to answer this question, it is necessary to define what it is that we mean by *privacy*, and what sort of *utility* we hope to achieve with our algorithms. In this thesis, we will use *differential privacy* as defined by Dwork et al. [DMNS06, Dwo06, CDM$^+$05, DN04] as our notion of privacy, together with several definitions of utility for different problems.

Even before defining the words *privacy* and *usefulness* however, it is intuitively clear that these two goals are at odds with one another. Our algorithms could simply output random bits, independently of the private data set, and this should be "privacy preserving" for any reasonable definition of privacy, but should provide no utility, for any reasonable definition of utility. Conversely, our algorithm could simply publish the entire private data set. This provides the user with full utility, but clearly does not preserve privacy in any sense of the word. Because the twin goals of privacy and usefulness are at odds, the fundamental question in the field of privacy preserving algorithms is how they must be traded off against each other, and this is the question we will consider in this thesis.

1.1 Overview of the Thesis

1.1.1 Releasing Statistical Information Privately

The main portion of this thesis concerns the problem of releasing statistical information about a dataset privately. We will be mainly concerned with a useful class of queries known as *predicate queries*, although all of our results hold more generally for the class of *linear queries*. Informally speaking, a predicate query specifies a property that may or may not be possessed by individuals in the dataset. The corresponding predicate query asks what fraction of individuals in the dataset satisfy that property. In addition to being a natural class of queries, the ability to accurately answer polynomially many predicate queries is sufficient to learn in the Statistical Query learning model of Kearns [Kea98].

An immediate question is: for given privacy constraints how many predicate queries can be answered privately, and to what accuracy? Prior work [DMNS06] has shown that for fixed privacy and accuracy constraints, it was possible to answer linearly many queries in the size of the database. Can we answer more?

In Chapter 3, we answer this question in the affirmative. We show that if the mechanism knows the queries ahead of time, it is information theoretically possible to answer *exponentially* many arbitrary predicate queries over a discrete domain. In fact, it can be possible to answer many more: we show that with a sufficiently large data set, we can answer every query in any class of polynomially bounded VC-dimension. We also show that this dependence on the VC-dimension of the predicate class is necessary, and optimal up to constant factors, and that there is no private mechanism for even simple predicate classes over continuous domains. Finally, we give specialized private-release mechanisms for the query classes of axis-aligned rectangle queries and large-margin halfspaces. These specialized mechanisms are much more computationally efficient than our general release mechanism. This work originally appeared in STOC 2008, and is joint with Avrim Blum and Katrina Ligett [BLR08].

In Chapter 4, we ask whether it is possible to achieve the results of Chapter 3 if the mechanism does *not* know the queries ahead of time, but must instead privately answer queries interactively as they arrive. We show that up to a small degradation in the parameters, the answer is *yes*. By showing how to correlate answer perturbations between queries online, we give the first mechanism for answering superlinearly many queries in the interactive model of database privacy. We also show how to implement our online mechanism in time polynomial in the size of the predicate domain, at the expense of trading our worst-case utility guarantees for average case guarantees. This run time is essentially optimal for a mechanism which can answer generic predicate queries (even in the offline model), due

to a cryptographic lower bound of Dwork et al. [DNR+09]. This work originally appeared in STOC 2010, and is joint with Tim Roughgarden. [RR10]

In Chapter 5, we generalize the results of chapters 3 and 4 from classes of predicate queries to the more general classes of *linear queries*. Although VC-dimension is no longer an appropriate measure of query complexity for linear queries, we show that *fat-shattering dimension*, which characterizes the agnostic learnability of real valued functions, is. We show polynomially related lower and upper bounds on the magnitude of noise that must be added to linear queries from some class C in terms of the fat-shattering dimension of C.

In Chapter 6, we turn our attention to query release in a more restrictive model of data privacy, in which there is no central and trusted database administrator. In this local model, originally studied by Kasiviswanathan et al. [KLN+08], individuals maintain control of their own private data, and interact with any release mechanism only in a differentially private manner. This model might be vastly preferable from the centralized model from a privacy point of view: even if there is (currently) a suitable trusted party to act as database administrator, aggregations of large amounts of private data are vulnerable to theft, and even absent theft, there is no guarantee that the database will be held to the same privacy standards in perpetuity. Unfortunately, we show that predicate query release is severely restricted in the local model. We show that no mechanism operating over a d dimensional domain can usefully release a superpolynomial (in d) number of *monotone conjunctions* to subconstant error. Monotone conjunctions are an extremely simple class of queries, and so our lower bound holds also for any class that generalizes them (such as conjunctions, disjunctions, halfspaces, polynomially sized decision trees, etc.) Unfortunately, the small number of existing techniques for *computationally efficient* data release can be implemented in the local modal, and so this result suggests that in order to develop efficient predicate query release mechanisms for more expressive classes, fundamentally new techniques will have to be developed to overcome the local-privacy barrier.

1.1.2 Performing Combinatorial Optimization Privately

In Chapter 7, we study private algorithms for tasks that go beyond answering statistical queries, and demand combinatorial, rather than numeric outputs.

Consider the problem of outfitting a small number of hospitals with specialized HIV treatment centers so that the average commute time of patients is small. This can be modeled as an instance of the k-median problem, for which good approximation algorithms are known. However the problem instance is defined in part by the home addresses of

HIV patients, which is sensitive information, and the output of standard approximation algorithms can reveal a great deal about their inputs. In this case, data privacy is a crucial design goal, which motivates the question of how we must trade of the quality of the solution with the constraint of differential privacy.

In contrast to previous work in differential privacy, which largely concentrated on answering *numeric* queries, in this work we initiate a systematic study of approximation algorithms for discrete optimization problems that preserve differential privacy. We show that many combinatorial optimization problems indeed have good approximation algorithms which preserve differential privacy, even in cases where it is impossible to preserve cryptographic definitions of privacy while computing non-trivial approximations to even the *value* of the optimal solution, let alone the solution itself. We consider the k-median problem, weighted and unweighted vertex cover and set cover, min-cut, and the recently introduced submodular maximization problem *combinatorial public projects* (CPP). This last result is notable because the CPP problem was recently shown to be inapproximable to polynomial multiplicative factors by any efficient and *truthful* mechanism [PSS08]. Our differentially private algorithm yields an *approximately truthful* mechanism which achieves a logarithmic additive approximation.

For each of the results we consider, we prove tight information theoretic upper and lower bounds, as well as give *efficient* differentially private algorithms. This work originally appeared in SODA 2010, and is joint work with Anupam Gupta, Katrina Ligett, Frank McSherry, and Kunal Talwar [GLM$^+$10].

Chapter 2

Background, Definitions, and Related Work

2.1 Privacy: Why and How

Large data-sets of sensitive personal information are becoming increasingly common – no longer are they the domain only of census agencies: now hospitals, social networks, insurance companies, and search engines all possess vast amounts of sensitive data. These organizations face legal, financial, and moral pressure to make their data available to the public, but also face legal, financial, and moral pressure to protect the identities of the individuals in their dataset. Formal guarantees trading off privacy and utility are therefore of the utmost importance.

Unfortunately, the history of data privacy is littered with the failed attempts at data anonymization and ad-hoc fixes to prevent specific attacks. The most egregious of these attempts at anonymization simply scrub the dataset of obviously identifiable information (such as names), and release the remaining data in its complete form. These failed attempts at data privacy exhibit a common characteristic: a failure of imagination on the part of the data curator to account for what auxiliary information an attacker may have available, in addition to the "privately" released data set. It is this observation that will motivate differential privacy: a guarantee that provably holds independent of the attacker's auxiliary information.

One of the first published attacks on publicly available data was due to Sweeney [S+02]. She attacked a dataset administered by the Group Insurance Commission, which consisted of medical records of employees of the state of Massachusetts. All "identifi-

able information" (such as names, social security numbers, phone numbers, etc.) had been removed from this dataset, and so its release had been deemed safe. It contained only seemingly innocuous information such as birth date, gender, and zip code. However, there was another easily accessible data set that linked birthdate, gender, and zip code to names: the Massachusetts voter registration list. These three fields proved sufficiently unique in combination that they were often sufficient to *uniquely* identify an individual in the dataset. In particular, by cross referencing these two datasets, Sweeney was able to identify the medical records of William Weld, the sitting Governor of Massachusetts.

The method employed by Sweeney is an example of what is known as a *linkage attack*: in which identifying information is associated with anonymous records by cross-referencing them with some additional source of information. Other examples abound: One of the most well publicized is Narayanan and Shmatikov's attack on the Netflix Prize data set [NS08]. The Netflix competition began in 2006, with a 1 million dollar prize promised to the team that could improve the algorithm that Netflix uses to recommend movies to customers, by 10%. To facilitate the competition, Netflix provided a data set consisting of over $100,000,000$ movie ratings from over $480,000$ users. Each user was assigned an "anonymous" identifier, and was associated with each of the movies she had rated, together with her ratings, and the dates that the ratings were provided. Narayanan and Shmatikov carried out a more complex linkage attack than had Sweeney. They used the IMDB database of movie reviews – publicly available and associated with personal identifying information – to crossreference with the Netflix data set, allowing approximate matches for date. The end result was however the same: they were able to re-identify a substantial fraction of the users in the NetFlix data set.

Not all privacy breaches are due to linkage attacks. Another high profile privacy breach came when AOL released a purportedly "anonymized" collection of search logs in 2006 [BZ06]. The data included the search queries of users, together with which link from the search results was selected by the user. For privacy, user names were removed, and replaced by random numeric identifiers. Three days after the data set was released, the New York Times identified and interviewed a user from the data set. In order to re-identify the user, the Times did not need any outside data set: the search queries themselves (terms such as the name of a town, a last name, etc.) provided enough information to identify their owner.

Each of these cases involved some ad-hoc privacy measure which failed to be private because the database administrator neglected to anticipate a particular kind of attack (utilizing an outside database, or the semantic content of the released data set itself). Subsequently, various definitions of privacy were developed to defend against specific types of attacks. For example, Sweeney proposed k-*anonymity* as a privacy definition to defend

against the linkage attack that she used to compromise William Weld's medical records [S$^+$02]. Informally, a released data set is k-anonymous if there are no released tuples of characteristics that correspond to fewer than k individuals. Although this prevents linkage attacks from specifically identifying individuals, it does not prevent an adversary from learning private information about an individual. For example, if an individual's data can be linked to k medical records, all of which correspond to cancer patients, then the adversary has still learned that the individual in question has cancer. Even in cases in which the data is not quite so obvious (the linked records do not all have the same diagnoses, for example), they can still provide substantial information about an individual when combined with prior beliefs.

Because of the history of successful attacks that make use of unanticipated sources of outside data, it seems prudent to work with a definition of privacy that provides a guarantee independent of whatever prior knowledge an attacker may have about the database or about the population from which it is drawn. A moment's thought reveals that any such definition cannot constrain merely the object that is eventually released, but must constrain the algorithm by which that object is constructed. In particular, any definition of privacy that is resilient to arbitrary auxiliary information must only allow randomized algorithms to perform 'private' data release. Any deterministic algorithm that computes a non-constant function reveals with its output a great deal about its input: in particular, it rules out certain inputs with certainty, which may substantially sharpen an adversaries posterior beliefs over the input.

But what should a formal constraint of 'privacy' encode? In 1977, Dalenius proposed a criteria for privacy in statistical databases: that nothing about an individual should be learnable from the database that cannot be learned without the database [Dal77]. Unfortunately, as shown by Dwork, this type of privacy cannot be achieved [Dwo06]. In addition to a formalization of this statement, Dwork also provides the following illustrative example giving the intuition behind the obstacle to achieving Dalenial privacy, which is the presence of *auxiliary information* which might be available to the adversary.

> Suppose one's exact height were considered a highly sensitive piece of information, and that revealing the exact height of an individual were a privacy breach. Assume that the database yields the average heights of women of different nationalities. An adversary who has access to the statistical database and the auxiliary information "Terry Gross is two inches shorter than the average Lithuanian woman" learns Terry Gross' height, while anyone learning only the auxiliary information, without access to the average heights, learns very little. [Dwo06]

A notable aspect of this example is that this "privacy breach" occurs even if Terry Gross is not in the database. It seems intuitively that a proper notion of privacy should not be violated by a release of data that is completely independent from the individual in question.

Differential Privacy, which we describe in the next section, provides exactly what we would like: an information theoretic guarantee on the distribution from which the private outputs of an algorithm are generated, that is independent of whatever auxiliary information an attacker may have available to him. It is an extremely strong definition of privacy, and the challenge in the coming chapters will be to demonstrate algorithms that satisfy it while still obtaining a high degree of utility.

2.2 Differential Privacy

Informally, differential privacy encodes the constraint that small changes in the private data set should only cause small changes in the output distribution of an algorithm acting on the data. What this means is that while we hope that the outputs of our algorithms are informative functions of their inputs, they should not be strongly correlated to any *particular* element of the input. This definition has an appealing interpretation in terms of the ability of a data set curator to collect private information from a privacy-aware population: when asked for private data, an individual may be unwilling to provide it, because she fears some consequence of the use of this data. However, if the data is used in a differentially-private manner, then she can be assured that *any* event that she is afraid of is (almost) no more likely to occur if she provides her data, as compared to if she does not provide her data. No matter what she fears, therefore, she should be willing to provide her private information to a data set curator, given only a small positive incentive for doing so (monetary or otherwise). We shall now define differential privacy precisely.

Let $\mathcal{D} \in \mathcal{X}^n$ denote a *data set* or *database* (we will use these terms interchangeably), which we will typically regard as an unordered set of n elements from some abstract domain $\mathcal{X}$. We should view each element of $\mathcal{D}$ as corresponding to a different individual, whose privacy must be preserved. For any two data sets $\mathcal{D}$ and $\mathcal{D}'$, we write $\mathcal{D} \triangle \mathcal{D}'$ to denote their symmetric difference. For any two data sets $\mathcal{D}$ and $\mathcal{D}'$, we say that $\mathcal{D}$ and $\mathcal{D}'$ are *neighboring data sets* if $|\mathcal{D} \triangle \mathcal{D}'| = 1$ – that is, if they differ only in a single user's data.

An *algorithm* or *mechanism* $\mathcal{M} : \mathcal{X}^* \to \triangle(\mathcal{R})$ is a function that maps databases $\mathcal{D} \in \mathcal{X}^*$ to probability distributions over some range $\mathcal{R}$: that is, it is a randomized algorithm that acts on databases, and randomly outputs some element from $\mathcal{R}$ according to a probability distribution that depends on $\mathcal{D}$.

Definition 1. *A mechanism $\mathcal{M}$ preserves (α, τ)-differential privacy if for every pair of neighboring databases $\mathcal{D}$ and $\mathcal{D}'$, and for every set of outcomes $S \subseteq \mathcal{R}$, $\mathcal{M}$ satisfies:*

$$\Pr[\mathcal{M}(\mathcal{D}) \in S] \leq e^{\alpha} \Pr[\mathcal{M}(\mathcal{D}') \in S] + \tau$$

If $\tau = 0$, we say that the mechanism preserves α-differential privacy.

α-differential privacy is stricter than (α, τ)-differential privacy: it constrains *every* event to change in probability by only a small multiplicative factor for every element in $\mathcal{D}$ that is changed. In particular, this applies to events with zero probability, and so an α-differentially private mechanism must place positive probability on every event in its range. In contrast, (α, τ)-differential privacy gives us the freedom to violate strict α-differential privacy for low probability events: in particular, we may have some events have zero probability for certain databases, but not for others. This ability will often prove useful from an algorithm design perspective; however, since we do not want to weaken the definition of α-differential privacy substantially, we will generally only be interested in extremely small values of τ: ideally *negligibly* small – smaller than any inverse polynomial in the parameters of the problem. In contrast, we will generally think of α as being a small constant. It will be meaningful in some settings to speak of subconstant values for α, but it is not hard to see that no meaningful utility guarantees are possible for values of α smaller than $O(\frac{1}{n})$.[1]

Differential privacy is a robust definition, because it has some particularly nice composability properties:

Fact 1 (Composability)**.** *If $\mathcal{M}_1$ preserves α_1-differential privacy, and $\mathcal{M}_2$ preserves α_2-differential privacy, then $\mathcal{M}(D) = (\mathcal{M}_1(D), \mathcal{M}_2(D))$ preserves $\alpha_1 + \alpha_2$-differential privacy.*

Fact 2 (Post-Processing)**.** *If $\mathcal{M} : \mathcal{X}^* \to \mathcal{R}$ preserves (α, τ)-differential privacy, and $f : \mathcal{R} \to \mathcal{R}'$ is any arbitrary (database independent) function, then $f(\mathcal{M}) : \mathcal{X}^* \to \mathcal{R}'$ preserves (α, τ)-differential privacy.*

A key concept in the literature on data privacy is that of the ℓ_1-sensitivity of a function f, as defined by Dwork et al. [DMNS06]. This is essentially a Lipschitz condition on f:

[1] If we have $\alpha = o(1/n)$, then even after replacing every element in some data set $\mathcal{D}$, an α-differentially private mechanism could not change the probability of any outcome by more than a factor of $\exp(n \cdot o(1/n)) = \exp(o(1)) \approx 1$ – that is, it would have approximately the same output distribution for *every* input database, and so would not give any useful information about $\mathcal{D}$.

Definition 2 (ℓ_1-sensitivity [DMNS06])**.** *Given a function $f : \mathcal{X}^* \to \mathbb{R}^k$, the ℓ_1-sensitivity of f is:*

$$S(f) = \max_{\mathcal{D},\mathcal{D}'} ||f(\mathcal{D}) - f(\mathcal{D}')||_1$$

where the maximum is taken over all pairs of neighboring *databases.*

Dwork, McSherry, Nissim, and Smith [DMNS06] defined what we will refer to as the *Laplace* mechanism. The Laplace mechanism makes use of the Laplace distribution, which is a symmetric exponential distribution:

Definition 3 (The Laplace Distribution)**.** *The Laplace Distribution (centered at 0) with scale b is the distribution with probability density function:*

$$\textit{Lap}(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$$

We will sometimes write Lap(b) to denote the Laplace distribution with scale b.

Definition 4 (The Laplace Mechanism [DMNS06])**.** *Given any function $f : \mathcal{X}^* \to \mathbb{R}^k$, the Laplace mechanism is defined as:*

$$\mathcal{M}_L(\mathcal{D}, f(\cdot), \alpha) = f(\mathcal{D}) + (Y_1, \ldots, Y_k)$$

where Y_i are independent draws from Lap$(S(f)/\alpha)$

Dwork et al. prove that the Laplace mechanism preserves α-differential privacy – we give a proof here as well for completeness in the case of nonadaptive queries, but Dwork et al. prove the theorem for adaptive queries as well [DMNS06]:

Theorem 1 ([DMNS06])**.** *The Laplace mechanism preserves α-differential privacy.*

Proof. Let D and D' be any pair of neighboring databases, and let $f(\cdot)$ be some function $f : \mathcal{X}^* \to \mathbb{R}^k$. Let p_D denote the probability density function of $\mathcal{M}_L(D, f)$, and let $p_{D'}$ denote the probability density function of $\mathcal{M}_L(D', f)$. We compare the two at some

arbitrary point $x \in \mathbb{R}^k$

$$\begin{aligned}
\frac{p_D(x)}{p_{D'}(x)} &= \prod_{i=1}^{k} \left(\frac{\exp(-\frac{\alpha|f(D)_i - x_i|}{S(f)})}{\exp(-\frac{\alpha|f(D')_i - x_i|}{S(f)})} \right) \\
&= \prod_{i=1}^{k} \exp\left(\frac{\alpha(|f(D')_i - x_i| - |f(D)_i - x_i|)}{S(f)} \right) \\
&\leq \prod_{i=1}^{k} \exp\left(\frac{\alpha|f(D)_i - f(D')_i|}{S(f)} \right) \\
&= \exp\left(\frac{\alpha \cdot ||f(D) - f(D')||_1}{S(f)} \right) \\
&\leq \exp(\alpha)
\end{aligned}$$

where the first inequality follows from the triangle inequality, and the last follows from the definition of sensitivity and the fact that D and D' are neighboring databases. That $\frac{p_D x}{p_{D'}(x)} \geq \exp(-\alpha)$ follows by symmetry. □

The Laplace mechanism is useful for privately answering numerically valued queries. Suppose we would like to answer non-numeric queries? McSherry and Talwar [MT07] define the *exponential mechanism*, which can be used to privately output objects from some arbitrary domain while preserving differential privacy. Given some arbitrary range $\mathcal{R}$, the exponential mechanism is defined with respect so some quality function $q : \mathcal{X}^* \times \mathcal{R} \to \mathbb{R}$, which maps database/output pairs to quality scores. We should interpret this intuitively as a statement that fixing a database $\mathcal{D}$, the user would prefer the mechanism to output some element of $\mathcal{R}$ with as high a quality score as possible.

Definition 5 (The Exponential Mechanism [MT07])**.** *The exponential mechanism $\mathcal{M}_E(\mathcal{D}, q)$ selects and outputs an element $r \in \mathcal{R}$ with probability proportional to $\exp(\frac{\alpha q(\mathcal{D},r)}{2S(q)}) \cdot \mu(r)$, where $\mu(r)$ is some base measure over $\mathcal{R}$ (which we will typically take to be uniform, and so can be ignored).*

McSherry and Talwar prove that the exponential mechanism preserves α-differential privacy. It is important to note that the exponential mechanism can define a complex distribution over a large arbitrary domain, and so it may not be possible to implement the exponential mechanism efficiently.

Theorem 2 ([MT07])**.** *The exponential mechanism preserves α-differential privacy.*

Proof. For clarity, we assume the range $\mathcal{R}$ of the exponential mechanism is finite, but this is not necessary. As in all differential privacy proofs, we consider the ratio of the probability that an instantiation of the exponential mechanism outputs some element $r \in \mathcal{R}$ on two neighboring databases D and D'.

$$\begin{aligned}
\frac{\Pr[\mathcal{M}_E(D,q)=r]}{\Pr[\mathcal{M}_E(D,q)=r]} &= \frac{\left(\frac{\exp(\frac{\alpha q(\mathcal{D},r)}{2S(q)})\cdot\mu(r)}{\sum_{r'\in\mathcal{R}}\exp(\frac{\alpha q(\mathcal{D},r')}{2S(q)})\cdot\mu(r')}\right)}{\left(\frac{\exp(\frac{\alpha q(\mathcal{D}',r)}{2S(q)})\cdot\mu(r)}{\sum_{r'\in\mathcal{R}}\exp(\frac{\alpha q(\mathcal{D}',r')}{2S(q)})\cdot\mu(r')}\right)} \\
&= \left(\frac{\exp(\frac{\alpha q(\mathcal{D},r)}{2S(q)})}{\exp(\frac{\alpha q(\mathcal{D}',r)}{2S(q)})}\right)\cdot\left(\frac{\sum_{r'\in\mathcal{R}}\exp(\frac{\alpha q(\mathcal{D}',r')}{2S(q)})\cdot\mu(r')}{\sum_{r'\in\mathcal{R}}\exp(\frac{\alpha q(\mathcal{D},r')}{2S(q)})\cdot\mu(r')}\right) \\
&= \exp\left(\frac{\alpha(q(\mathcal{D},r')-q(\mathcal{D}',r'))}{2S(q)}\right)\cdot\left(\frac{\sum_{r'\in\mathcal{R}}\exp(\frac{\alpha q(\mathcal{D}',r')}{2S(q)})\cdot\mu(r')}{\sum_{r'\in\mathcal{R}}\exp(\frac{\alpha q(\mathcal{D},r')}{2S(q)})\cdot\mu(r')}\right) \\
&\leq \exp\left(\frac{\alpha}{2}\right)\cdot\exp\left(\frac{\alpha}{2}\right)\cdot\left(\frac{\sum_{r'\in\mathcal{R}}\exp(\frac{\alpha q(\mathcal{D},r')}{2S(q)})\cdot\mu(r')}{\sum_{r'\in\mathcal{R}}\exp(\frac{\alpha q(\mathcal{D},r')}{2S(q)})\cdot\mu(r')}\right) \\
&= \exp(\alpha)
\end{aligned}$$

Similarly, $\frac{\Pr[\mathcal{M}_E(D,q)=r]}{\Pr[\mathcal{M}_E(D,q)=r]} \geq \exp(-\alpha)$ by symmetry. □

This concludes a summary of the basic building blocks of differential privacy that we will use in this thesis. There is however by now an extensive and growing literature, which we will review in the next section.

2.3 Predicate Queries

A large fraction of this thesis will be devoted to designing algorithms for answering *predicate queries*, and their generalizations. Informally, predicate queries specify some quality that a database entry might or might not posses. The predicate query asks what fraction of the elements in the database posses the given quality. We say that a *predicate* f is a boolean function $f : \mathcal{X} \to \{0,1\}$. We say that $x \in \mathcal{X}$ *satisfies* f if $f(x) = 1$. Given a database $\mathcal{D}$, we abuse notation and write that the predicate query:

$$f(\mathcal{D}) = \frac{|\{x \in \mathcal{D} : f(x) = 1\}|}{|\mathcal{D}|}$$

is the query which determines what fraction of the database $\mathcal{D}$ satisfies predicate f. We will be interested in mechanisms which preserve differential privacy, but also produce outputs which are useful for answering predicate queries. Formally:

Definition 6. *A mechanism $\mathcal{M}$ is (ϵ, δ)-useful if for every database $\mathcal{D}$ and every set of predicate queries $\mathcal{C} = \{f_1, \ldots, f_k\}$, the mechanism $\mathcal{M}(\mathcal{D})$ produces answers $a_1, \ldots, a_k$ such that with probability at least $1 - \delta$:*

$$\max_{f_i \in \mathcal{C}} |f_i(\mathcal{D}) - a_i| \leq \epsilon$$

That is, an (ϵ, δ)-useful mechanism answers every query it is supplied with correctly (up to additive error ϵ) with high probability $(1 - \delta)$. We may also speak of mechanisms which are (ϵ, δ)-useful with respect to a particular type of predicate query, rather than for arbitrary sets of predicate queries. We will typically think of ϵ as a small constant (and never smaller than $O(1/\sqrt{n})$), whereas we should think of δ as being inverse-polynomially small in n.

We note that the definition of usefulness does not specify *how* the mechanism represents the answers it supplies. For example, the mechanism could simply output the numeric values $(a_1, \ldots, a_k)$. However, it could also output some datastructure from which the answer to each query can be efficiently decoded, or a concise representation of all of the answers. One example of such a concise representation comes in the form of a synthetic database $\mathcal{D}' \in \mathcal{X}^n$. Such a database can encode the answers to queries in a very natural way: $a_i = f_i(\mathcal{D}')$.

We will sometimes make the distinction between *interactive* and *non-interactive* mechanisms. An interactive mechanism receives predicate queries online, and must answer each query before seeing the next query. A non-interactive mechanism receives all queries up front before it needs to provide any answers. We note that the Laplace mechanism can be used to interactively answer predicate queries:

Observation 3. *If $f(\cdot)$ is a predicate query, the ℓ_1 sensitivity of $f(\mathcal{D})$ is $S(f) \leq \frac{1}{|\mathcal{D}|}$. If $g(\mathcal{D}) = (f_1(\mathcal{D}), \ldots, f_k(\mathcal{D}))$ is the concatenation of k predicate queries, then $S(g) \leq \frac{k}{|\mathcal{D}|}$.*

This observation is immediate: a predicate query is simply a normalized count of the database elements that satisfy the predicate, and this count can be incremented by at most 1 by adding a single element to the database. The sensitivity of a concatenation of k queries can be at most the sum of their individual sensitivities, by the definition of the ℓ_1 norm.

We may of course use the Laplace mechanism to answer k predicate queries adaptively: the Laplace mechanism adds independent noise drawn from $\mathrm{Lap}(k/(|\mathcal{D}|\alpha))$ to each answer.

2.4 Related Work in Differential Privacy

Recent work on theoretical guarantees for data privacy was initiated by Dinur and Nissim [DN03]. The notion of differential privacy, developed in a series of papers [BDMN05, Dwo06, DKM$^+$06, DMNS06], separates issues of privacy from issues of outside information by defining privacy as indistinguishability of neighboring databases. This captures the notion that (nearly) anything that can be learned if your data is included in the database can also be learned without your data. This notion of privacy ensures that users have very little incentive to withhold their information from the database. The connection between data privacy and incentive-compatibility was recently formalized by McSherry and Talwar [MT07].

Dinur and Nissim studied a setting in which the private database consisted of bits held by different users, and the data analyst wished to know the sums of random subsets of these bits. They showed that there cannot be any private mechanism that answers n such queries with error $o(\sqrt{n})$, because an adversary could use any such mechanism to reconstruct a $1-o(1)$ fraction of the original database, a condition which they called *blatant non-privacy*. This result was strengthened by several subsequent papers [DMT07, DY08, KRSU10].

Dwork et al. gave the original definition of differential privacy, as well as the Laplace mechanism, which is capable of answering any k "low sensitivity" queries (including linear queries) up to error $O(k)$. A more refined analysis of the relationship between the laplace mechanism and function sensitivity was later given by [NRS07].

McSherry and Talwar introduced the exponential mechanism, and observed that differential privacy could be viewed as a solution concept in game theory, which in particular implied approximate truthfulness [MT07]. They applied the exponential mechanism to obtain differentially private, approximately revenue maximizing auctions.

Kasiviswanathan et al. [KLN$^+$08] applied the exponential mechanism of McSherry and Talwar to the problem of selecting a function from some class $\mathcal{C}$ which best classifies points in a private database $\mathcal{D}$. They proved that any concept class $\mathcal{C}$ that has a PAC learning algorithm also has a privacy preserving learning algorithm, at least if one ignores computational constraints. In Chapter 3, we adapt the approach of [KLN$^+$08] to show that it is possible to release a synthetic database which may be used to usefully answer queries about $\mathcal{D}$ for every predicate query in some set $\mathcal{C}$, where the privacy parameter α grows proportionally to the VC-dimension of $\mathcal{C}$. Since VC-DIM$(\mathcal{C}) \leq \log |\mathcal{C}|$ for any set of predicate queries $\mathcal{C}$, we may rephrase this result as follows, to compare it to the Laplace mechanism: For a fixed database size n and accuracy and privacy targets ϵ and α, the instantiation of the exponential mechanism used in Chapter 3 may answer $O(2^{n(\alpha\epsilon^3/\log|\mathcal{X}|)})$

queries, which is nearly optimal in its dependence on n and $|\mathcal{X}|$.

Since the mechanism of Chapter 3 is an instantiation of the exponential mechanism, its naive implementation does not run in time polynomial in the parameters of the problem. Unlike the Laplace mechanism, it also does not operate in the interactive setting – that is, it requires that the set of functions $\mathcal{C}$ be provided up front, before any queries are answered. Dwork et al. [DNR+09] showed that modulo cryptographic assumptions, no mechanism with run time polynomial in $\log |\mathcal{X}|$ or $\log k$ can achieve such a bound for all predicate queries. They also gave a mechanism with run time polynomial in $|\mathcal{X}|$ and k that for a fixed database size and accuracy target, could preserve (α, τ)-differential privacy while answering $O(2^{\log^2(n(\alpha\epsilon^2/\log|\mathcal{X}|))})$ predicate queries. This mechanism is *non-interactive*. For constant values of ϵ and α, this is fewer queries than the mechanism of Chapter 3 by a superpolynomial factor, but we note that the dependence on ϵ is better (and optimal), and so can provide a superior bound for sufficiently small values of ϵ.

Hardt and Talwar [HT10] gave a non-interactive mechanism for answering linear queries (which slightly generalize predicate queries), as well as a lower bound, which is optimal in terms of certain parameters of the problem. The bounds in [HT10] are incomparable to ours: loosely the bounds of [HT10] are better when the total number of queries is smaller than the size of the database, and our bounds are better when the total number of queries is significantly larger than the size of the database. We make this explicit and generalize our bounds to linear queries in Chapter 5.

Nissim, Raskhodnikova, and Smith consider the *smooth sensitivity* of functions, which is a refined analysis of the global sensitivity of functions [NRS07]. Using this tool, they give a private algorithm for k-means clustering, which is the first example of a differentially private algorithm for a combinatorial optimization task. In Chapter 7, we use different techniques to give differentially private approximation algorithms for a large number of tasks. Independently of our work, Feldman et al. [FFKN09] also give a differentially private algorithm for the k-medians problem, one of the problems that we consider in Chapter 7.

There is a large and growing literature on differential privacy, and the work mentioned here is in no way intended to be a complete representation. We will review the most relevant related work at the start of each chapter, and the interested reader is directed towards several excellent surveys: [Dwo08, DS10].

Chapter 3

Answering Exponentially Many Predicate Queries

3.1 Introduction

As large-scale collection of personal information becomes easier, the problem of database privacy is increasingly important. In many cases, we might hope to learn useful information from sensitive data (for example, we might learn a correlation between smoking and lung cancer from a collection of medical records). However, for legal, financial, or moral reasons, administrators of sensitive datasets might not want to release their data. If those with the expertise to learn from large datasets are not the same as those who administer the datasets, what is to be done?

There have been a series of lower bound results [DN03, DMNS06, DMT07] that suggested that non-interactive databases (or interactive databases that can be queried a linear number of times) cannot accurately answer all queries, or an adversary will be able to reconstruct all but a $1-o(1)$ fraction of the original database exactly (obviously a very strong violation of privacy). These lower bounds in fact only preclude answering queries to very high accuracy, but as a result, attention in the privacy community shifted to the design of interactive mechanisms that answer only a sublinear number of queries. However, since these mechanisms may only answer a sublinear number of queries *in total* (not per user), after which point they must be destroyed, this limits their practicality in situations where the number of queries that might be asked is comparable to or larger than the number of entries in the database.

In this chapter, motivated by learning theory, we propose the study of privacy-

preserving mechanisms that are useful for all queries in a particular class (such as all conjunctive queries or all halfspace queries). In particular, we focus on predicate queries of the form, "what fraction of the database entries satisfy predicate φ?" and say that a sanitized output is useful for a class C if the answers to all queries in C have changed by at most some $\pm\epsilon$. Building on the techniques of Kasiviswanathan et al. [KLN$^+$07], we show that for discretized domains, for any concept class with polynomial VC-dimension, it is possible to release differential-privacy-preserving databases that are simultaneously useful for all queries in the concept class. The algorithm may not be computationally efficient in general, though we do have a computationally efficient method for range queries over a finite interval with bounded precision.

Unfortunately, we show that for non-discretized domains, under the above definition of usefulness, it is impossible to publish a differentially private non-interactive database that is useful for even quite simple classes such as interval queries. We next show how, under a natural relaxation of the usefulness criterion, one can release information that can be used to usefully answer (arbitrarily many) halfspace queries while satisfying privacy. In particular, instead of requiring that useful mechanisms answer each query approximately correctly, we allow our algorithm to produce an answer that is approximately correct *for some nearby query*. This relaxation is motivated by the notion of large-margin separators in learning theory [AB99, Vap98, SS02]; in particular, queries with no data points close to the separating hyperplane must be answered accurately, and the allowable error more generally is a function of the fraction of points close to the hyperplane.

We also introduce a new concept, *distributional privacy*, which makes explicit the notion that when run on a database drawn from a distribution, privacy-preserving mechanisms should reveal only information about the underlying distribution, and nothing else. A database privatization mechanism satisfies distribution privacy if for all distributions $\mathcal{D}$ over database points, drawing an entirely new database from $\mathcal{D}$ does not change the probability of any outcome of the privatization mechanism by more than some small amount.

We show that distributional privacy is a strictly stronger guarantee than differential privacy by showing that any mechanism that satisfies distributional privacy also satisfies differential privacy, but that there are some functions that can be answered accurately while satisfying differential privacy, and yet reveal information about the particular database (although not about any particular database element) that is not "distributional."

We also show, in section 3.8.2, a lower bound on the noise that must be added to privately answered predicate queries.

3.1.1 Related Work

Prior work on interactive mechanisms has implied a number of impossibility results for non-interactive mechanisms. Dinur and Nissim [DN03] show that if a database administrator answers all "subset sum" with less than $o(\sqrt{n})$ noise, this would allow an adversary to reconstruct a $1 - o(1)$ fraction of the database[1]. Dwork et al. [DMT07] show that even if the privacy-preserving mechanism is allowed to answer a small constant fraction of queries arbitrarily, if the remaining queries still are perturbed with $o(\sqrt{n})$ noise, an adversary can still reconstruct the database.

Dwork et al. [DMNS06] introduced the Laplace mechanism, and showed that it was possible to answer queries interactively with noise proportional to their ℓ_1 sensitivity. For predicate queries, this comes with the caveat that for nontrivial accuracy and privacy guarantees, the mechanism can only answer a sublinear number of queries in total, and then no further information about the database can ever be released. Blum et al. [BDMN05] consider a model of learning and show that concept classes that are learnable in the statistical query (SQ) model are also learnable from a polynomially sized dataset accessed through an interactive differential-privacy-preserving mechanism. We note that such mechanisms still may only answer a fixed number of queries in total.

Most similar to the work in this Chapter is the recent work of Kasiviswanathan et al. [KLN$^+$07]. Kasiviswanathan et al. study what can be learned privately when what is desired is that the hypothesis output by the learning algorithm satisfies differential privacy. They show that in a PAC learning model in which the learner has access to the private database, ignoring computational constraints, anything that is PAC learnable is also privately PAC learnable. We build upon the technique in their paper to show that in fact, it is possible to privately release a dataset that is simultaneously useful for any function in a concept class of polynomial VC-dimension. This resolves an open question posed by [KLN$^+$07] about whether a VC-dimension analogue of Occam's razor holds in their private learning model. Kasiviswanathan et al. also study several restrictions on learning algorithms, show separation between these learning models, and give efficient algorithms for learning particular concept classes.

In this work, we study non-interactive database release mechanisms, which may be used to answer an unlimited number of queries. Blum et al. [BDMN05] consider running machine learning algorithms on datasets that are accessed through interactive privacy-preserving mechanisms. In contrast, we show how to release data sets from which one can usefully learn the values of all functions in restricted concept classes.

[1]Consider a database that is a collection of private bits $b_1, \ldots, b_n$, where $b_i \in \{0, 1\}$. A subset sum query specifies a subset of $[n]$ and corresponds to the sum of the corresponding bits.

3.1.2 Motivation from Learning Theory

From a machine learning perspective, one of the main *reasons* one would want to perform statistical analysis of a database in the first place is to gain information about the population from which that database was drawn. In particular, a fundamental result in learning theory is that if one views a database as a collection of random draws from some distribution $\mathcal{D}$, and one is interested in a particular class C of boolean predicates over examples, then a database D of size $\tilde{O}(\text{VCDIM}(C)/\epsilon^2)$ is sufficient so that with high probability, for *every* query $q \in C$, the proportion of examples in D satisfying q is within $\pm\epsilon$ of the true probability mass under $\mathcal{D}$ [AB99, Vap98].[2] Our main result can be viewed as asking how much larger does a database D have to be in order to do this in a privacy-preserving manner: that is, to allow one to (probabilistically) construct an output $\hat{D}$ that accurately approximates $\mathcal{D}$ with respect to all queries in C, and yet that reveals no extra information about database D.[3] In fact, our notion of distributional privacy (Section 3.7) is motivated by this view. Note that since the Laplace mechanism can handle arbitrary queries of this form so long as only $o(n)$ are requested, our objective is interesting only for classes C that contain $\Omega(n)$, or even exponentially in n many queries. We will indeed achieve this (Theorem 4), since $|C| \geq 2^{\text{VCDIM(C)}}$.

3.1.3 Organization

We present essential definitions in Section 3.2. In Section 3.3, we show that, ignoring computational constraints, one can release sanitized databases over discretized domains that are useful for *any* concept class with polynomial VC-dimension. We then, in Section 3.4, give an efficient algorithm for privately releasing a database useful for the class of interval queries. We next turn to the study of halfspace queries over $\mathbb{R}^d$ and show in Section 3.5 that, without relaxing the definition of usefulness, one cannot release a database that is privacy-preserving and useful for halfspace queries over a continuous domain. Relaxing our definition of usefulness, in Section 3.6, we give an algorithm that in polynomial time, creates a sanitized database that usefully and privately answers all halfspace queries. We

[2]Usually, this kind of uniform convergence is stated as empirical error approximating true error. In our setting, we have no notion of an "intrinsic label" of database elements. Rather, we imagine that different users may be interested in learning different things. For example, one user might want to learn a rule to predict feature x_d from features $x_1, \ldots, x_{d-1}$; another might want to use the first half of the features to predict a certain boolean function over the second half.

[3]Formally, we only care about $\hat{D}$ approximating D with respect to C, and want this to be true no matter how D was constructed. However, if D *was* a random sample from a distribution $\mathcal{D}$, then D will approximate $\mathcal{D}$ and therefore $\hat{D}$ will as well.

present an alternative definition of privacy and discuss its relationship to differential privacy in Section 3.7. In section 3.8.2, we give a separation of interactive and non-interactive databases for predicate queries.

3.2 Definitions

In this section, we briefly recall some relevant definitions.

For a database D, let M be a database access mechanism. For an interactive mechanism, we will say that $M(D, Q)$ induces a distribution over outputs for each query Q. For a non-interactive mechanism, we will say that $M(D)$ induces a distribution over outputs.

We say that an interactive database access mechanism M satisfies α-*differential privacy* if for all neighboring databases D_1 and D_2 (differing in only a single element), for all queries Q, and for all outputs x,

$$\Pr[M(D_1, Q) = x] \leq e^{\alpha} \Pr[M(D_2, Q) = x].$$

We say that a non-interactive database sanitization mechanism A satisfies α-*differential privacy* if for all neighboring databases D_1 and D_2, and for all sanitized outputs $\widehat{D}$,

$$\Pr[M(D_1) = \widehat{D}] \leq e^{\alpha} \Pr[M(D_2) = \widehat{D}].$$

In Section 3.7, we propose an alternate definition of privacy, distributional privacy, and show that it is strictly stronger than differential privacy. For simplicity, however, in the main body of the paper, we use the standard definition, differential privacy. All of these proofs can be adapted to the distributional privacy notion.

We recall that the Laplace mechanism $\mathcal{M}_L(\mathcal{D}, f(\cdot), \alpha)$ is a database access mechanisms that can answer any low-sensitivity query while preserving differential privacy. However, lower bounds of Dinur and Nissim [DN03] and Dwork et al. [DMNS06] imply that such mechanisms can only answer a sublinear number of queries on any database. Note that these mechanisms can only answer a sublinear number of queries *in total*, not per user.

We propose to construct database access mechanisms whose results can be released to the public, and so can necessarily be used to answer an arbitrarily large number of queries. We seek to do this while simultaneously preserving privacy. Note that differential privacy precludes the use of several naive techniques: for example, simply outputting a subsample of our private database will not preserve differential privacy – since the presence of

an individual in the database would give him a nonzero probability of appearing in the subsample, whereas in his absence, his probability of appearing in the subsample would drop to 0. We cannot hope to be able to usefully answer arbitrary queries. We instead seek to answer restricted classes of queries while preserving "usefulness," which we define as follows:

Definition 7 (Usefulness definition 1)**.** *A database mechanism A is (ϵ, δ)*-useful *for queries in class C if with probability $1 - \delta$, for every $Q \in C$ and every database D, for $\widehat{D} = A(D)$, $|Q(\widehat{D}) - Q(D)| \leq \epsilon$.*

Mechanisms satisfying this definition must output databases that appear to be similar to D with respect to the functions in C only – with respect to other queries, D and $\widehat{D}$ may be completely different.

3.3 General release mechanism

In this section we show that (ignoring computational considerations) it is possible to release a non-interactive database useful for any concept class with polynomial VC-dimension, while preserving α-differential privacy, given an initial database of polynomial size. Our use of the exponential mechanism is inspired by its use by Kasiviswanathan et al. [KLN$^+$07].

The results in this section are presented for answering boolean valued predicate queries over the boolean hypercube. However, this is easily generalized to real valued functions over some arbitrary domain X: $f : X \to [0, 1]$, where the desired quantity is the average value of f over the elements of the private database. We handle this level of generality in Chapter 5.

Theorem 4. *For any class of functions C, and any database $D \subset \{0, 1\}^d$ such that*

$$|D| \geq O\left(\frac{d\textit{VCDIM}(C)\log(1/\epsilon)}{\epsilon^3\alpha} + \frac{\log(1/\delta)}{\alpha\epsilon}\right)$$

we can output an (ϵ, δ)-useful database $\widehat{D}$ that preserves α-differential privacy. Note that the algorithm is not necessarily efficient.

We give an (inefficient) algorithm that outputs a sanitized database $\widehat{D}$ of size $\widetilde{O}(\text{VCDIM}(C)/\epsilon^2)$. We note that the size of the output database is independent of the size of our initial database. This is sufficient for (ϵ, δ)-usefulness because the set of all databases of this size forms an ϵ-cover with respect to C of the set of all possible databases.

Lemma 1 ([AB99, Vap98])**.** *Given any database D there exists a database $\widehat{D}$ of size $m = O(\textit{VCDIM}(C)\log(1/\epsilon)/\epsilon^2)$ such that $\max_{h \in C} |h(D) - h(\widehat{D})| < \epsilon/2$.*

Proof. By [AB99], a uniformly random sample $\widehat{D}$ from D of size $m = O(\text{VCDIM}(C)\log(1/\epsilon)/\epsilon^2)$ will have the property that $\max_{h \in C} |h(D) - h(\widehat{D})| < \epsilon/2$ with constant probability. In particular, *some* database with this property must exist. □

We note that we use the above sample complexity result merely existentially: we do not actually ever take a sample from our private database, and releasing such a sample would be a privacy violation.

We now instantiate the exponential mechanism of McSherry and Talwar [MT07] for the problem of outputting databases as follows:

Definition 8. *For any function $q : ((\{0,1\}^d)^n \times (\{0,1\}^d)^m) \to \mathbb{R}$ and input database D, the exponential mechanism outputs each database $\widehat{D}$ with probability proportional to $e^{q(D,\widehat{D})\alpha n/2}$.*

Theorem 5 ([MT07])**.** *The exponential mechanism preserves (α)-differential privacy.*

Proof of Theorem 4. We use the exponential mechanism and define our quality function q to be:

$$q(D, \widehat{D}) = -\max_{h \in C} \left| h(D) - h(\widehat{D}) \right|$$

Note that $GS_q = 1/n$. In order to show that this mechanism satisfies (ϵ, δ)-usefulness, we must show that it outputs some database $\widehat{D}$ with $q(D, \widehat{D}) \geq -\epsilon$ except with probability δ.

Any output database $\widehat{D}$ with $q(D, \widehat{D}) \leq -\epsilon$ will be output with probability at most proportional to $e^{-\alpha\epsilon n/2}$. There are at most 2^{dm} possible output databases, and so by a union bound, the probability that we output any database $\widehat{D}$ with $q(D, \widehat{D}) \leq -\epsilon$ is at most proportional to $2^{dm}e^{-\alpha\epsilon n/2}$.

Conversely, we know by Lemma 1 that there exists some $\widehat{D} \in (\{0,1\}^d)^m)$ such that $q(D, \widehat{D}) \geq -\epsilon/2$, and therefore that such a database is output with probability at least proportional to $e^{-\alpha\epsilon n/4}$.

Let A be the event that the exponential mechanism outputs some database $\widehat{D}$ such that $q(D, \widehat{D}) \geq -\epsilon/2$. Let B be the event that the exponential mechanism outputs some database $\widehat{D}$ such that $q(D, \widehat{D}) \leq -\epsilon$. We must show that $\Pr[A]/\Pr[B] \geq (1 - \delta)/\delta$.

$$\begin{aligned}\frac{\Pr[A]}{\Pr[B]} &\geq \frac{e^{-\alpha\epsilon n/4}}{2^{dm}e^{-\alpha\epsilon n/2}} \\ &= \frac{e^{\alpha\epsilon n/4}}{2^{dm}}\end{aligned}$$

Setting this quantity to be at least $1/\delta > (1-\delta)/\delta$, we see that it is sufficient to take

$$\begin{aligned}n &\geq \frac{4}{\epsilon\alpha}\left(dm + \ln\frac{1}{\delta}\right) \\ &\geq O\left(\frac{d\textbf{VCDIM}(C)\log(1/\epsilon)}{\epsilon^3\alpha} + \frac{\log(1/\delta)}{\alpha\epsilon}\right).\end{aligned}$$

This result extends in a straightforward manner to the case of any discretized database domain, not just a boolean space. □

Theorem 4 shows that a database of size $\widetilde{O}(\frac{d\text{VCDIM}(C)}{\epsilon^3\alpha})$ is sufficient in order to output a set of points that is ϵ-useful for a concept class C, while simultaneously preserving α-differential privacy. If we were to view our database as having been drawn from some distribution $\mathcal{D}$, this is only an extra $\widetilde{O}(\frac{d}{\epsilon\alpha})$ factor larger than what would be required to achieve ϵ-usefulness with respect to $\mathcal{D}$, even without any privacy guarantee! In fact, as we will show in Theorem 17, it is impossible to release a database that is $o(1/\sqrt{n})$-useful for the class of parity functions while preserving privacy, and so a dependence on ϵ of at least $\Omega(1/\epsilon^2)$ is necessary.

The results in this section only apply for discretized database domains, and may not be computationally efficient. We explore these two issues further in the remaining sections of this chapter.

3.4 Interval queries

In this section we give an *efficient* algorithm for privately releasing a database useful for the class of interval queries over a discretized domain, given a database of size only polynomial in our privacy and usefulness parameters. We note that our algorithm is easily extended to the class of axis-aligned rectangles in d dimensional space for d a constant; we present the case of $d = 1$ for clarity.

Consider a database D of n points in $[0,1]$ in which the entries are discretized to b bits of precision; our bounds will be polynomial in b (in Corollary 9 we show some discretization is necessary). Given $a_1 \leq a_2$, both in $[0,1]$, let I_{a_1,a_2} be the indicator function corresponding to the interval $[a_1, a_2]$. That is:

$$I_{a_1,a_2}(x) = \begin{cases} 1, & a_1 \leq x \leq a_2; \\ 0, & \text{otherwise.} \end{cases}$$

Definition 9. *An interval query $Q_{[a_1,a_2]}$ is defined to be*

$$Q_{[a_1,a_2]}(D) = \sum_{x \in D} \frac{I_{a_1,a_2}(x)}{|D|}.$$

Note that $GS_{Q_{[a_1,a_2]}} = 1/n$, and we may answer interval queries while preserving α-differential privacy by adding noise proportional to $\text{Lap}(1/(\alpha n))$.

Given a database D, we will use α'-differential privacy preserving interval queries to perform a binary search on the interval $[0,1]$ and partition it into sub-intervals containing probability mass in the range $[\epsilon_1/2 - \epsilon_2, \epsilon_1/2 + \epsilon_2]$. Because of the discretization, the depth of this search is at most b. We will then output a dataset that has $(\epsilon_1/2) \cdot n$ points in each of these intervals. Because we have constructed this dataset using only a small number of privacy preserving queries, its release will also preserve privacy, and it will be (ϵ, δ)-useful for the class of interval queries with an appropriate choice of parameters. Finally, this simple mechanism is clearly computationally efficient.

Theorem 6. *With $\alpha' = (\epsilon\alpha)/4b$, $\epsilon_1 = (\epsilon/2)$ and $\epsilon_2 = (\epsilon^2/8)$, the above mechanism preserves α-differential privacy while being (ϵ, δ)-useful for the class of interval queries given a database of size:*

$$|D| \geq O\left(\frac{b(\log b + \log(1/\epsilon\delta))}{\alpha\epsilon^3}\right)$$

Proof. We first bound the number of privacy preserving queries our algorithm makes. It finally produces $2/\epsilon_1$ intervals. Since D is defined over a discretized space, we can identify each interval with the at most b queries on its path through the binary search procedure, and so we will make a total of at most $2b/\epsilon_1 = 4b/(\epsilon)$ α'-differential privacy preserving queries. Since the differential-privacy parameter composes, with $\alpha' = (\epsilon\alpha)/4b$, our algorithm indeed preserves α differential privacy.

Suppose that the binary search procedure indeed returns intervals each containing probability mass in the range $[\epsilon_1/2 - \epsilon_2, \epsilon_1/2 + \epsilon_2]$. Any query will intersect at most

two of these intervals only partially. In the worst case, this introduces $\epsilon_1 = \epsilon/2$ error to the query ($\epsilon_1/2$ error from each interval that partially overlaps with the query). Since each query can only overlap at most $2/\epsilon_1$ intervals, and each interval contains a probability mass that deviates from the true probability mass in D by at most ϵ_2, this introduces an additional $2\epsilon_2/\epsilon_1 = \epsilon/2$ error, for a total error rate $\leq \epsilon$. Therefore, to complete the proof, we only need to bound the size of D necessary such that the probability that any of the $2b/\epsilon_1$ privacy preserving queries returns an answer that deviates from the true answer (in D) by more than ϵ_2 is less than δ. Let us call this event FAILURE. Since the event that any single query has error rate $\geq \epsilon_2$ is $\Pr[\text{Lap}(1/(\alpha' n)) \geq \epsilon_2] \leq e^{\alpha' \epsilon_2 n}$, this follows from a simple union bound:

$$\Pr[\text{FAILURE}] \leq \frac{2b}{\epsilon_1} e^{(-\epsilon\alpha/4b)\epsilon_2 n} \leq \delta.$$

Solving, we find

$$n \geq \frac{4b(\log 2b) + \log(1/\epsilon_1\delta)}{\alpha\epsilon\epsilon_2} = O\left(\frac{b(\log b + \log(1/\epsilon\delta))}{\alpha\epsilon^3}\right)$$

is sufficient. □

We note that although the class of intervals (and more generally, low dimensional axis-aligned rectangles) is a simple class of functions, it nevertheless contains exponentially (in b) many queries, and so it is not feasible to simply ask all possible interval queries using an interactive mechanism.

While it is true that intervals (and low dimensional axis-aligned rectangles) have constant VC-dimension and polynomial ϵ-cover size, we can trivially extend the above results to the class of unions of t intervals by dividing ϵ by t and answering each interval separately. This class has VC-dimension $O(t)$ and exponentially large ϵ-cover size, and the running time of our algorithm grows by only a factor of t. This yields the following theorem:

Theorem 7. *There exist classes of functions C with VC-Dimension t for which there are differentially private release mechanisms $\mathcal{M}$ that have running time poly(t) and data requirements poly$(t, \alpha, 1/\epsilon, \log 1/\delta, \log X)$ such that $\mathcal{M}$ is (ϵ, δ) useful with respect to C, and preserves α-differential privacy.*

3.5 Lower bounds

Could we possibly modify the results of Sections 3.4 and 3.3 to hold for non-discretized databases? Suppose we could usefully answer an arbitrary number of queries in some simple concept class C representing interval queries on the real line (for example, "How many points are contained within the following interval?") while still preserving privacy. Then, for any database containing single-dimensional real valued points, we would be able to answer median queries with values that fall between the $1/2 - \delta, 1/2 + \delta$ percentile of database points by performing a binary search on D using A (where $\delta = \delta(\epsilon)$ is some small constant depending on the usefulness parameter ϵ). However, answering such queries is impossible while guaranteeing differential privacy. Unfortunately, this would seem to rule out usefully answering queries in simple concept classes such as halfspaces and axis-aligned rectangles, that are generalizations of intervals.

Theorem 8. *No mechanism A can answer median queries M with outputs that fall between the $1/2-k, 1/2+k$ percentile with positive probability on any real valued database D, while still preserving α-differential privacy, for $k < 1/2$ and any α.*

Proof. Consider real valued databases containing elements in the interval $[0, 1]$. Let $D_0 = (0, \ldots, 0)$ be the database containing n points with value 0. Then we must have $\Pr[A(D_0, M) = 0] > 0$. Since $[0, 1]$ is a continuous interval, there must be some value $v \in [0, 1]$ such that $\Pr[A(D_0, M) = v] = 0$. Let $D_n = (v, \ldots, v)$ be the database containing n points with value v. We must have $\Pr[A(D_n, M) = v] > 0$. For $1 < i < n$, let $D_i = (\underbrace{0, \ldots, 0}_{n-i}, \underbrace{v, \ldots, v}_{i})$. Then we must have for some i, $\Pr[A(D_i, M) = v] = 0$ but $\Pr[A(D_{i+1}, M) = v] > 0$. But since D_i and D_{i+1} differ only in a single element, this violates differential privacy. □

Corollary 9. *No mechanism can be (ϵ, δ)-useful for the class of interval queries, nor for any class C that generalizes interval queries to higher dimensions (for example, halfspaces, axis-aligned rectangles, or spheres), while preserving α-differential privacy, for any $\epsilon = o(n)$ and any α.*

Proof. Consider any real valued database containing elements in the interval $[0, 1]$. If A is (ϵ, δ)-useful for interval queries and preserves differential privacy, then we can construct a mechanism A' that can answer median queries with outputs that fall between the $1/2 - k, 1/2 + k$ percentile with positive probability while preserving differential privacy. By Theorem 8, this is impossible. A' simply computes $\widehat{D} = A(D)$, and performs binary search on $\widehat{D}$ to find some interval $[0, a]$ that contains $n/2 \pm \epsilon$ points. Privacy is preserved

since we only access D through A, which by assumption preserves differential privacy. With positive probability, all interval queries on $\widehat{D}$ are correct to within $\pm\epsilon$, and so the binary search can proceed. Since $\epsilon = o(n)$, the result follows. □

We may get around the impossibility result of Corollary 9 by relaxing our definitions. One approach is to discretize the database domain, as we do in Sections 3.3 and 3.4. Another approach, which we take in Section 3.6, is to relax our definition of usefulness:

Definition 10 (Usefulness definition 2)**.** *A database mechanism A is $(\epsilon, \delta, \gamma)$-*useful *for queries in class C according to some metric d if with probability $1 - \delta$, for every $Q \in C$ and every database D, $|Q(A(D)) - Q'(D)| \leq \epsilon$ for some $Q' \in C$ such that $d(Q, Q') \leq \gamma$.*

We remark that this relaxed definition of usefulness corresponds to the idea of margin in learning theory. In the next section, we instantiate this to give a mechanism for large-margin halfspace queries. We may alternatively view it as follows: we wish to receive an estimate of the fraction of positive points according to a halfspace classifier that must correctly classify any point that is "very" positive or negative (i.e. far from the halfspace), but is given the freedom to misclassify points that are very near the halfspace. This might be an acceptable guarantee if we believe that our data is drawn from a noisy source, and have low confidence for the classification of points that lie very near the target halfspace.

3.6 Answering Halfspace Queries

Here, we consider databases that contain n elements in $\mathbb{R}^d$. In this section, we show how to efficiently release information useful (according to definition 10) for the class of halfspace queries for any constant $\gamma > 0$. Throughout this section, we assume without loss of generality that the database points are scaled into the unit sphere. Additionally, when we project the points into a lower-dimensional space, we rescale them to the unit sphere. A halfspace query specifies a hyperplane in $\mathbb{R}^d$ and asks how many points fall above it:

Definition 11. *Given a database $D \subset \mathbb{R}^d$ and unit length $y \in \mathbb{R}^d$, a halfspace query H_y is*

$$H_y(D) = \frac{|\{x \in D : \sum_{i=1}^{d} x_i \cdot y_i \geq 0\}|}{|D|}.$$

The assumption that halfspaces pass through the origin is without loss of generality since we can view translated halfspaces as passing through the origin in a space of dimension $d + 1$.

In this section, we give an algorithm that is $(\epsilon, \delta, \gamma)$-useful for the class of halfspace queries. For a point x we will write $\hat{x}$ for the normalization $x/||x||_2$. We define the distance between a point x and a halfspace H_y by $d(x, H_y) = |\hat{x} \cdot y|$. For convenience, we define the distance between two halfspaces H_{y_1} and H_{y_2} to be the sin of the angle between y_1 and y_2; by a slight abuse of notation, we will denote this by $d(y_1, y_2)$. In particular, for a point x and two halfpaces H_{y_1} and H_{y_2}, $d(x, H_{y_1}) \leq d(x, H_{y_2}) + d(y_1, y_2)$. If $d(y_1, y_2) \leq \gamma$ we say that H_{y_1} and H_{y_2} are γ-*close*. Given a halfspace H_{y_1}, our goal is to output a value v such that $|v - H_{y_2}(D)| < \epsilon$ for some H_{y_2} that is γ-close to H_{y_1}. Equivalently, we may arbitrarily count or not count any point $x \in D$ such that $d(x, H_{y_1}) \leq \gamma$. We note that γ is similar to the notion of margin in machine learning, and that even if H_{y_1} and H_{y_2} are γ-close, this does not imply that $H_{y_1}(D)$ and $H_{y_2}(D)$ are close, unless most of the data points are outside a γ margin of H_{y_1} and H_{y_2}.

We circumvent the halfspace-lower bound of Corollary 9 by considering a class of *discretized* halfspaces:

Definition 12. *A halfspace query H_y is* b-discretized *if for each $i \in [d]$, y_i can be specified with b-bits. Let C_b be the set of all b-discretized halfspaces in $\mathbb{R}^d$.*

We first summarize the algorithm, with the parameters to be specified later. Our use of random projections is similar to that in the work of Indyk and Motwani [IM98] on approximate nearest neighbor queries.

Our algorithm performs m random projections $P_1, \ldots, P_m$ of the data onto $\mathbb{R}^k$. A random projection of n points from $\mathbb{R}^d$ to $\mathbb{R}^k$ is defined as follows:

Definition 13. *A random projection P_i from $\mathbb{R}^d$ to $\mathbb{R}^k$ is defined by a $d \times k$ random matrix M_i with entries chosen independently and uniformly at random from $\{-1, 1\}$. We write the projection of point $x \in \mathbb{R}^d$ as $P_i(x) = (1/\sqrt{k})x \cdot M_i$. We write the projection of a database $D \in (\mathbb{R}^d)^n$ as $P_i(D) = \{P_i(x) : x \in D\}$.*

For each projected database $P_i(D)$ we ask $O(1/\gamma^{k-1})$ privacy-preserving canonical halfspace queries. To answer a halfspace query H_y, for each projection P_i, we consider $H_{P_i(y)}$ and associate with it the answer of the closest canonical halfspace in that projection. Finally, we return the median value of these queries over all m projections.

Theorem 10 (Johnson-Lindenstrauss [DG99, BBV06])**.** *Consider a random projection P of a point x and a halfspace H_y onto a random k-dimensional subspace. Then*

$$\Pr[|d(x, H_y) - d(P(x), H_{P(y)})| \geq \gamma/4] \leq 2e^{-((\gamma/16)^2 - (\gamma/16)^3)k/4}.$$

That is, projecting x and H_y significantly changes the distance between the point and the halfspace with only a small probability.

We choose k such that the probability that projecting a point and a halfspace changes their distance by more than $\gamma/4$ is at most $\epsilon_1/4$. Solving, this yields

$$k \geq \frac{4\ln(8/\epsilon_1)}{(\gamma/16)^2 - (\gamma/16)^3}.$$

Given a halfspace H_y and a point x, we say that a projection P makes a mistake relative to x and H_y if $d(x, H_y) \geq \gamma/4$, but $\text{sign}(x \cdot y) \neq \text{sign}(P(x) \cdot P(y))$. We have chosen k such that the expected fraction of mistakes relative to any halfspace H_y in any projection P is at most $\epsilon_1/4$. By Markov's inequality, therefore, the probability that a projection makes more than $\epsilon_1 n$ mistakes relative to a particular halfspace is at most $1/4$.

The probability δ_1 that more than $m/2$ projections make more than $\epsilon_1 n$ mistakes relative to *any* discretized halfspace is at most $2^{bd}e^{-m/12}$ by a Chernoff bound and a union bound. Solving for m, this gives

$$m \geq 12\left(\ln\left(\frac{1}{\delta_1}\right) + \ln(2)bd\right).$$

For each projection P_i, we select a $(3/4)\gamma$-net of halfspaces N_i, such that for every vector $y_1 \in \mathbb{R}^k$ corresponding to halfspace H_{y_1}, there exists a halfspace $H_{y_2} \in N_i$ such that $d(y_1, y_2) \leq (3/4)\gamma$. We note that $|N_i| = O(1/\gamma^{k-1})$. For each projection P_i and for each $H_y \in N_i$, we record the value of

$$v_y^i = \mathcal{M}_L(P_i(D), H_y, \alpha/(m|N_i|)).$$

We note that since we make $m|N_i|$ queries in total, these queries preserve α-differential privacy.

Taking a union bound over the halfspaces in each N_i, we find that the probability δ_2 that any of the v_y^i differ from $H_y(P_i(D))$ by more than ϵ_2 is at most $m \cdot O(1/\gamma)^{k-1}e^{-(\epsilon_2 n\alpha)/(mO(1/\gamma^{k-1}))}$. Solving for n, we find that

$$\begin{aligned} n &\geq \frac{\log(1/\delta_2) + \log m + (k-1)\log(1/\gamma) + mO(1/\gamma)^{k-1}}{\epsilon_2\alpha} \\ &= O\Bigg(\frac{1}{\epsilon_2\alpha}\Bigg(\log(1/\delta_2) + \log\log 1/\delta_1 + \log bd + \log(1/\epsilon_1) \\ &\qquad + (\log 1/\delta_1 + bd)(1/\epsilon_1)^{(4\log(1/\gamma))/((\gamma/16)^2-(\gamma/16)^3)}\Bigg)\Bigg). \end{aligned}$$

To respond to a query H_y, for each projection P_i we first compute

$$H_{y'_i} = \operatorname*{argmin}_{H_{y'_i} \in N_i} d(P(y), y'_i).$$

We recall that by construction, $d(P(y), y'_i) \leq (3/4)\gamma$. We then return the median value from the set $\{v^i_{y'_i} : i \in [m]\}$.

Theorem 11. *The above algorithm is $(\epsilon, \gamma, \delta)$-useful while maintaining α-differential privacy for a database of size poly$(\log(1/\delta), 1/\epsilon, 1/\alpha, b, d)$ and running in time poly$(\log(1/\delta),\ 1/\epsilon, 1/\alpha, b, d)$, for constant γ.*

Proof. Above, we set the value of m such that for any halfspace query H_y, with probability at most δ_1, no more than an ϵ_1 fraction of the points have the property that they are outside of a γ margin of H_y but yet their projections are within a $(3/4)\gamma$ margin of $H_{P_i(y)}$, where i is the index of the median projection. Therefore, answering a query $H_{y'}$, where y' is $(3/4)\gamma$-close to $P_i(y)$, only introduces ϵ_1 error. Moreover, we have chosen n such that except with probability δ_2, the privacy-preserving queries introduce no more than an additional ϵ_2 error. The theorem follows by setting $\epsilon_1 = \epsilon_2 = \epsilon/2$ and $\delta_1 = \delta_2 = \delta/2$, and setting n, m, and k as above. □

3.7 Distributional Privacy

In this section, we propose an alternative definition of privacy motivated by the idea that private databases are actually collections of individuals drawn from some underlying population. Our definition formalizes the idea that we want algorithms that give us information about the underlying population, without providing any information that is specific to the sample. Our definition will guarantee that with high probability, the outcomes of a private mechanism when run on two databases sampled from the same underlying population should be indistinguishable. For each mechanism, we will require that this property hold for every population simultaneously, which will give us a stronger guarantee than differential privacy.

We say that an interactive database mechanism A satisfies (α, β)-*distributional privacy* if for any underlying population of database elements $\mathcal{D}$, with probability $1 - \beta$, two databases D_1 and D_2 consisting of n elements drawn *without replacement* from $\mathcal{D}$, for any query Q and output x satisfies

$$\Pr[A(D_1, Q) = x] \leq e^{\alpha} \Pr[A(D_2, Q) = x].$$

Similarly, for non-interactive mechanisms, a mechanism A satisfies (α, β)-*distributional privacy* if for any underlying population of database elements D, with probability $1 - \beta$, two databases D_1 and D_2 consisting of n elements drawn *without replacement* from $\mathcal{D}$, and for all sanitized outputs $\widehat{D}$,

$$\Pr[A(D_1) = \widehat{D}] \leq e^{\alpha} \Pr[A(D_2) = \widehat{D}].$$

For example, suppose that a collection of hospitals in a region each treats a random sample of patients with disease X. Distributional privacy means that a hospital can release its data anonymously, without necessarily revealing which hospital the data came from. Actually, our main motivation is that this definition is particularly natural from the perspective of learning theory: given a sample of points drawn from some distribution $\mathcal{D}$, one would like to reveal no more information about the sample than is inherent in $\mathcal{D}$ itself.

We will typically think of β as being exponentially small, whereas α must be $\Omega(1/n)$ for A to be useful.

3.7.1 Relationship Between Definitions

It is not a priori clear whether either differential privacy or distributional privacy is a stronger notion than the other, or if the two are equivalent, or distinct. On the one hand, differential privacy only provides a guarantee when D_1 and D_2 differ in a single element,[4] whereas distributional privacy can provide a guarantee for two databases D_1 and D_2 that differ in all of their elements. On the other hand, distributional privacy makes the strong assumption that the elements in D_1 and D_2 are drawn from some distribution $\mathcal{D}$, and allows for privacy violations with some exponentially small probability β (necessarily: with some small probability, two databases drawn from the same distribution might nevertheless be statistically completely different). However, as we show, distributional privacy is a strictly stronger guarantee than differential privacy. For clarity, we prove this for interactive mechanisms only, but the results hold for non-interactive mechanisms as well, and the proofs require little modification.

Theorem 12. *If A satisfies (α, β)-distributional privacy for any $\beta = o(1/n^2)$, then A satisfies α-differential privacy.*

Proof. Consider any database D_1 drawn from domain R, and any neighboring database D_2 that differs from D_1 in only a single element $x \in R$. Let $\mathcal{D}$ be the uniform distribution over the set of $n + 1$ elements $D_1 \cup \{x\}$. If we draw two databases D_1', D_2' from $\mathcal{D}$, then

[4] We get $t\alpha$-differential privacy for D_1 and D_2 that differ in t elements.

with probability $2/n^2$ we have $\{D_1', D_2'\} = \{D_1, D_2\}$, and so if $\beta = o(1/n^2)$, we have with certainty that for all outputs $\widehat{D}$ and for all queries Q,

$$\Pr[A(D_1, Q) = \widehat{D}] \leq e^{\alpha} \Pr[A(D_2, Q) = \widehat{D}].$$

Therefore, A satisfies α-differential privacy. □

We now give an example to show that distributional privacy is a strictly stronger guarantee than differential privacy.

Definition 14. *Define the* mirrored mod m function *as follows:*

$$F_m(x) = \begin{cases} x \mod m, & \text{if } x \mod 2m < m; \\ 2m - x - 1 \mod m, & \text{otherwise.} \end{cases}$$

For a database $D \in \{0,1\}^n$, define the query

$$Q_m(D) = \frac{F_m(\sum_{i=0}^{n-1} D[i])}{|D|}.$$

Note that the global sensitivity of any query Q_m satisfies $GS_{Q_m} \leq 1/n$. Therefore, the mechanism A that answers queries Q_n by $A(D, Q_m) = Q_m(D) + Z$ where Z is drawn from $\text{Lap}(1/(\alpha n)) = \text{Lap}(GS_{Q_m}/\alpha)$ satisfies α-differential privacy, which follows from the results of Dwork et al. [DMNS06].

Theorem 13. *There exist mechanisms A with α-differential privacy, but without (α, β)-distributional privacy for any $\alpha < 1$, $\beta = o(1)$ (that is, for any meaningful values of α, β).*

Proof. Consider databases with elements drawn from $\mathcal{D} = \{0,1\}^n$ and the query $Q_{2/\alpha}$. As observed above, a mechanism A such that $A(D, Q_i) = Q_i(D) + Z$ for $Z \sim \text{Lap}(1/(\alpha n))$ has α-differential privacy for any i. Note however that with constant probability, two databases D_1, D_2 drawn from $\mathcal{D}$ have $|Q_{2/\alpha}(D_1) - Q_{2/\alpha}(D_2)| \geq 1/(\alpha n)$. Therefore, for any output x, we have that with constant probability,

$$\begin{aligned} \frac{\Pr[A(D_1, Q_{2/\alpha}) = x]}{\Pr[A(D_2, Q_{2/\alpha}) = x]} &= e^{-\alpha |Q_{2/\alpha}(D_1) - Q_{2/\alpha}(D_2)|} \\ &\leq e^{-\alpha n (\frac{1}{\alpha n})} \\ &= \frac{1}{e}. \quad \square \end{aligned}$$

□

Although there are simpler functions for which preserving distributional privacy requires more added noise than preserving differential privacy, the mirrored-mod function above is an example of a function for which it is possible to preserve differential privacy usefully, but yet impossible to reveal any useful information while preserving distributional privacy.

We emphasize that the order of quantifiers is important in the definition of distributional privacy: a distributionally private mechanism must provide a simultaneous guarantee for all possible underlying populations, and it is for this reason that distributional privacy is stronger than differential privacy. We also note that in order for distributional privacy to imply differential privacy, it is important that in the definition of distributional privacy, database elements are drawn from some underlying population $\mathcal{D}$ *without replacement*. Otherwise, for any non-trivial distribution, there is some database D_* that is drawn with probability at most $1/2^n$, and we may modify any distributional-privacy preserving mechanism A such that for every query Q, $A(D_*, Q) = D_*$, and for any $D_i \neq D_*$, $A(D_i, Q)$ behaves as before. Since this new behavior occurs with probability $\leq \beta$ over draws from D for $\beta = O(1/2^n)$, A still preserves distributional privacy, but no longer preserves differential privacy (which requires that the privacy guarantee hold for *every* pair of neighboring databases).

3.8 Lower Bounds

3.8.1 VC Dimension Lower Bounds

In this chapter, we have provided upper bounds on the sample complexity necessary for a useful and private data release mechanism in terms of the VC-dimension of the class of queries of interest. It is also possible to prove *lower bounds* on sample complexity in terms of VC-dimension. However, we defer the discussion of these lower bounds until Chapter 5, in which we prove these lower bounds in a more general setting.

3.8.2 Parity: a $1/\sqrt{n}$ lower bound

Dwork et al. [DMNS06] provide a separation between interactive and non-interactive differential-privacy preserving mechanisms for a class of queries that are not predicate queries. They also provide a separation between interactive and non-interactive "randomized-response" mechanisms for parity queries (defined below), which are predicate queries. "Randomized-response" mechanisms are a class of non-interactive mecha-

nisms that independently perturb each point in D and release the independently perturbed points. Here, we provide a small separation between interactive mechanisms and arbitrary non-interactive mechanisms that output datasets useful for parity queries. We prove this separation for mechanisms that preserve differential privacy—our separation therefore also holds for *distributional*-privacy preserving mechanisms.

Definition 15. *Given a database D containing n points in $\{-1,1\}^d$, and for $S \subseteq \{1,\ldots,d\}$, a* parity query *is given by*

$$PQ_S = \frac{\left|\{x \in D : \prod_{i \in S} x_i = 1\}\right|}{|D|}.$$

We show that for any non-interactive mechanism A that preserves α-differential privacy for $\alpha = \Omega(1/\text{poly}(n))$ and outputs a database $\widehat{D} = A(D)$, there exists some $S \subseteq \{1,\ldots,d\}$ such that $|PQ_S(\widehat{D}) - PQ_S(D)| = \Omega(1/\sqrt{n})$. This provides a separation, since for any S, $GS_{PQ_S} = 1/n$, and so for for any S, with high probability, the interactive mechanism $A(D,Q)$ of [DMNS06] satisfies $|A(D,PQ_S) - PQ_S(D)| = o(1/\sqrt{n})$ while satisfying α-differential privacy. This also shows that our bound from Theorem 4 cannot be improved to have a $o(1/\epsilon^2)$ dependence on ϵ.

We begin with the claim that given some database D consisting of n distinct points in $\{-1,1\}^d$, any non-interactive α-differential privacy preserving mechanism that outputs a sanitized database must with high probability output a database $\widehat{D}$ that differs from D on at least half of its points.

Claim 14. *If the non-interactive mechanism A preserves α-differential privacy for $\alpha = \Omega(1/\text{poly}(n))$, $\Pr[|\widehat{D} \cap D| \geq n/2] < 1/2$.*

We next present a few facts from discrete Fourier analysis.

Proposition 15. *For any function $h : \{-1,1\} \to \mathbb{R}$, we may express h as a linear combination of parity functions: $h(x) = \sum_{S \subseteq \{1,\ldots,d\}} \hat{h}(S)\chi_S(x)$, where $\chi_S(x) = \prod_{i \in S} x_i$. Moreover, the coefficients $\hat{h}(S)$ take values*

$$\hat{h}(S) = \frac{1}{2^d} \sum_{x \in \{-1,1\}^d} g(x)\chi_S(x).$$

Proposition 16 (Parseval's Identity)**.** *For any function $h : \{-1,1\} \to \mathbb{R}$,*

$$\frac{1}{2^d} \sum_{x \in \{-1,1\}^d} h(x)^2 = \sum_{S \subseteq \{1,\ldots,d\}} \hat{h}(S)^2.$$

Lemma 2. *For $D_1, D_2 \in (\{-1,1\}^d)^n$, if $|D_1 \cap D_2| \leq n/2$, then there exists $S \in \{1,\ldots,d\}$ such that $|PQ_S(D_1) - PQ_S(D_2)| = \Omega(1/\sqrt{n})$*[5].

Proof. Let $f(x) : \{-1,1\}^d \to \{0,1\}$ be the indicator function of D_1: $f(x) = 1 \Leftrightarrow x \in D_1$. Similarly, let $g(x) : \{-1,1\}^d \to \{0,1\}$ be the indicator function of D_2. By our hypothesis,

$$\sum_{x \in \{-1,1\}^n} |f(x) - g(x)| \geq n/2.$$

Therefore,

$$\begin{aligned} n/2 &\leq \sum_{x \in \{-1,1\}^d} |f(x) - g(x)| \\ &= \sum_{x \in \{-1,1\}^d} (f(x) - g(x))^2 \\ &= 2^d \sum_{S \subseteq \{1,\ldots,d\}} (\hat{f}(S) - \hat{g}(S))^2, \end{aligned}$$

where the first equality follows from the fact that f and g have range $\{0,1\}$, and the second follows from Parseval's identity and the linearity of Fourier coefficients. Therefore, there exists some $S \subseteq \{1,\ldots,d\}$ such that $(\hat{f}(S) - \hat{g}(S))^2 \geq n/2^{2d+1}$, and so $|\hat{f}(S) - \hat{g}(S)| \geq \sqrt{n}/(2^d\sqrt{2}))$. We also have

$$\begin{aligned} &\hat{f}(S) - \hat{g}(S) \\ &\quad = \frac{1}{2^d} \sum_{x \in \{-1,1\}^d} f(x)\chi_S(x) - \frac{1}{2^d} \sum_{x \in \{-1,1\}^d} g(x)\chi_S(x) \\ &\quad = \frac{1}{2^d} \sum_{x \in D_1} \chi_S(x) - \frac{1}{2^d} \sum_{x \in D_2} \chi_S(x) \\ &\quad = \frac{n}{2^{d-1}} (PQ_S(D_1) - PQ_S(D_2)). \end{aligned}$$

Therefore, $|(PQ_S(D_1) - PQ_S(D_2)| \geq \Omega(1/\sqrt{n})$, which completes the proof. □

Combining the Claim 14 and Lemma 2, we get our result:

Theorem 17. *For any non-interactive mechanism A that outputs a database $\hat{D}_1 = A(D_1)$ and preserves α-differential privacy for $\alpha = \Omega(1/poly(n))$, with probability $> 1/2$ there exists some $S \subseteq \{1,\ldots,d\}$ such that $|PQ_S(D_1) - PQ_S(\hat{D}_1)| = \Omega(1/\sqrt{n})$.*

[5]Note that we are implicitly assuming that $d = \Omega(\log n)$

3.9 Conclusions

In this work, we view the problem of database privacy through the lens of learning theory. This suggests both a new definition of privacy, distributional privacy (which we show is strictly stronger than differential privacy), and the idea that we can study usefulness relative to particular classes of functions. We are able to show that it is possible to release privacy-preserving databases that are useful for all queries over a discretized domain in a concept class with polynomial VC-dimension. We show that this discretization is necessary by proving that it is impossible to privately release a database that is useful for halfspace queries without relaxing our definition of usefulness, but we demonstrate an algorithm that does so efficiently under a small relaxation of this definition.

This work demonstrates that the existing very strong lower bounds for useful, privacy-preserving, non-interactive mechanisms are not insurmountable, but can be circumvented by a number of reasonable relaxations to the standard definitions. However, our paper leaves a number of important questions open. Prime among them is the question of *efficient* private data release—we have shown that information theoretically it is possible to release a database that is useful for any concept class with polynomial VC-dimension (under our original, strong definition of usefulness) while preserving differential privacy, but we know how to do this *efficiently* only for the simplest classes of functions. Is it possible to *efficiently* privately and usefully release a database for every concept class with polynomial VC-Dimension? Is it possible for the class of conjunctions? For the class of parity functions?

We have also only managed to achieve high accuracy private data release at the cost of *interactivity*. Our mechanisms must be given all of their queries up front. In the next chapter, however, we show that this restriction is not necessary by giving a mechanism which achieves roughly the same bounds, but in the online setting.

Chapter 4

Answering Queries Interactively: The Median Mechanism

4.1 Introduction

Achieving differential privacy requires "sufficiently noisy" answers [DN03]. For example, suppose we're interested in the result of a query f — a function from databases to some range — that simply counts the fraction of database elements that satisfy some predicate φ on X. A special case of a result in Dwork et al. [DMNS06] asserts that the following mechanism is α-differentially private: if the underlying database is D, output $f(D) + \Delta$, where the output perturbation Δ is drawn from the Laplace distribution Lap($\frac{1}{n\alpha}$) with density $p(y) = \frac{n\alpha}{2}\exp(-n\alpha|y|)$. Among all α-differentially private mechanisms, this one (or rather, a discretized analog of it) maximizes user utility in a strong sense [GRS09].

What if we care about more than a single one-dimensional statistic? Suppose we're interested in k predicate queries $f_1, \ldots, f_k$, where k could be large, even super-polynomial in n. A natural solution is to use an independent Laplace perturbation for each query answer [DMNS06]. To maintain α-differential privacy, the magnitude of noise has to scale linearly with k, with each perturbation drawn from Lap($\frac{k}{n\alpha}$). Put another way, suppose one fixes "usefulness parameters" ϵ, δ, and insists that the mechanism is (ϵ, δ)-*useful*, meaning that the outputs are within ϵ of the correct query answers with probability at least $1 - \delta$. This constrains the magnitude of the Laplace noise, and the privacy parameter α now suffers linearly with the number k of answered queries. This dependence limits the use of this mechanism to a sublinear $k = o(n)$ number of queries.

In Chapter 3 we showed that it is in principle possible to do better. Specifically, in

chapter 3 the exponential mechanism of McSherry and Talwar [MT07] is used to show that, for fixed usefulness parameters ϵ, δ, the privacy parameter α only has to scale *logarithmically* with the number of queries.[1] This permits simultaneous non-trivial utility and privacy guarantees even for an exponential number of queries. Moreover, this dependence on $\log k$ is necessary in every differentially private mechanism (see the lower bound in Chapter 5).

The mechanism in chapter 3 suffers from two drawbacks, however. First, it is *non-interactive*: it requires all queries $f_1, \ldots, f_k$ to be given up front, and computes (noisy) outputs of all of them at once.[2] By contrast, independent Laplace output perturbations can obviously be implemented interactively, with the queries arriving online and each answered immediately. There is good intuition for why the non-interactive setting helps: outperforming independent output perturbations requires correlating perturbations across multiple queries, and this is clearly easier when the queries are known in advance. Indeed, prior to the present work, no interactive mechanism better than independent Laplace perturbations was known.

Second, the mechanism in Chapter 3 is *highly inefficient*. It does not run even in time polynomial in the size of the database domain X, which we itself often view as exponential in the natural parameters of the problem. As a minimal goal, we might want a mechanism that has running time polynomial in n, k, and $|X|$; Dwork et al. [DNR$^+$09] prove that this is essentially the best one could hope for (under certain cryptographic assumptions). The mechanism in chapter 4 does not achieve even this because it requires sampling from a non-trivial probability distribution over an unstructured space of exponential size. Dwork et al. [DNR$^+$09] recently gave a (non-interactive) mechanism running in time polynomial in X that is better than independent Laplace perturbations, in that the privacy parameter α of the mechanism scales as $2^{\sqrt{\log k}}$ with the number of queries k (for fixed usefulness parameters ϵ, δ).

Very recently, Hardt and Talwar [HT10] gave upper and lower bounds for answering noninteractive linear queries which are tight in a related setting. These bounds are not tight in our setting, however, unless the number of queries is small with respect to the size of the database. When the number of queries is large, our mechanism actually yields error significantly less than required in general by their lower bound [3]. This is not a

[1] More generally, linearly with the VC dimension of the set of queries, which is always at most $\log_2 k$.

[2] Or rather, it computes a compact representation of these outputs in the form of a synthetic database.

[3] We give a mechanism for answering k "counting queries" with (unnormalized) coordinate-wise error $O(n^{2/3} \log k (\log |X|/\alpha)^{1/3})$. This is less error than required by their *lower* bound of roughly $\Omega(\sqrt{k \log(|X|/k)}/\alpha)$ unless $k \leq \tilde{O}((n\alpha/\log |X|)^{4/3})$. We can take k to be as large as $k = \tilde{\Omega}(2^{(n\alpha/\log |X|)^{1/3}})$, in which case our upper bound is a significant improvement – as are the upper bounds

contradiction, because when translated into the setting of [HT10], our database size n becomes a sparsity parameter that is not considered in their bounds.

4.1.1 Our Results

We define a new interactive differentially private mechanism for answering k arbitrary predicate queries, called the *median mechanism*.[4] The basic implementation of the median mechanism interactively answers queries $f_1, \ldots, f_k$ that arrive online, is (ϵ, δ)-useful, and has privacy α that scales with $\log k \log |X|$; see Theorem 18 for the exact statement. These privacy and utility guarantees hold even if an adversary can adaptively choose each f_i after seeing the mechanism's first $i - 1$ answers. *This is the first interactive mechanism for answering predicate queries with noise less than the Laplace mechanism, and its performance is close to the best possible even in the non-interactive setting.*

The basic implementation of the median mechanism does not run in time polynomial in $|X|$, and we give a polynomial time variant with a somewhat weaker utility guarantee. (The privacy guarantee is as strong as in the basic implementation.) This alternative implementation runs in time polynomial in n, k, and $|X|$, and satisfies the following (Theorem 22): for every sequence $f_1, \ldots, f_k$ of predicate queries, for all but a negligible fraction of input distributions, the efficient median mechanism is (ϵ, δ)-useful.

This is the first efficient mechanism with a non-trivial utility guarantee and polylogarithmic privacy cost, even in the non-interactive setting.

4.1.2 The Main Ideas

The key challenge to designing an interactive mechanism that outperforms the Laplace mechanism lies in determining the appropriate correlations between different output perturbations on the fly, without knowledge of future queries. It is not obvious that anything significantly better than independent perturbations is possible in the interactive setting.

Our median mechanism and our analysis of it can be summarized, at a high level, by three facts. First, among any set of k queries, we prove that there are $O(\log k \log |X|)$ "hard" queries, the answers to which completely determine the answers to all of the other queries (up to $\pm\epsilon$). Roughly, this holds because: (i) by a VC dimension argument, we can focus on databases over X of size only $O(\log k)$; and (ii) every time we answer a "hard"

of [BLR08] and [DNR+09].

[4] The privacy guarantee is (α, τ)-differential privacy for a negligible function τ; see Section **??** for definitions.

query, the number of databases consistent with the mechanism's answers shrinks by a constant factor, and this number cannot drop below 1 (because of the true input database). Second, we design a method to privately release an indicator vector which distinguishes between hard and easy queries online. We note that a similar private 'indicator vector' technique was used by Dwork et al. [DNR$^+$09]. Essentially, the median mechanism deems a query "easy" if a majority of the databases that are consistent (up to $\pm\epsilon$) with the previous answers of the mechanism would answer the current query accurately. The median mechanism answers the small number of hard queries using independent Laplace perturbations. It answers an easy query (accurately) using the median query result given by databases that are consistent with previous answers. A key intuition is that if a user knows that query i is easy, *then it can generate the mechanism's answer on its own*. Thus answering an easy query communicates only a single new bit of information: that the query is easy. Finally, we show how to release the classification of queries as "easy" and "hard" with low privacy cost; intuitively, this is possible because (independent of the database) there can be only $O(\log k \log |X|)$ hard queries.

Our basic implementation of the median mechanism is not efficient for the same reasons as for the mechanism in [BLR08]: it requires non-trivial sampling from a set of super-polynomial size. For our efficient implementation, we pass to *fractional* databases, represented as fractional histograms with components indexed by X. Here, we use the random walk technology of Dyer, Frieze, and Kannan [DFK91] for convex bodies to perform efficient random sampling. To explain why our utility guarantee no longer holds for every input database, recall the first fact used in the basic implementation: every answer to a hard query shrinks the number of consistent databases by a constant factor, and this number starts at $|X|^{O(\log k)}$ and cannot drop below 1. With fractional databases (where polytope volumes play the role of set sizes), the lower bound of 1 on the set of consistent (fractional) databases no longer holds. Nonetheless, we prove a lower bound on the volume of this set for almost all fractional histograms (equivalently probability distributions), which salvages the $O(\log k \log |X|)$ bound on hard queries for databases drawn from such distributions.

4.1.3 Related Work

The central questions in the study of private data release are first, "to what accuracy ϵ can a single query be answered while preserving α-differential privacy", and second, once satisfied with an accuracy ϵ, "how must the privacy cost α of ϵ-accurate data release scale with the number of queries answered?" The answer to the first question is already well understood (see [DMNS06, NRS07, GRS09]). In this work we address the second question.

This question has been studied from several perspectives, and an important distinction in the field has been the difference between *interactive* and *non-interactive* mechanisms. A non-interactive mechanism takes as input a set of queries (all at once), and then privately outputs either a set of answers, or a data structure from which the set of answers can be recovered[5]. An interactive mechanism receives and answers queries online, and the entire transcript of the mechanism must be privacy preserving.

The first differentially-private mechanism proposed was the Laplace mechanism of Dwork et al. [**?**]. The Laplace mechanism interactively answers queries by computing their answers, and adding to them noise drawn from the Laplace distribution with magnitude proportional to the *sensitivity* of the query. The Laplace mechanism has the advantage of being simple, efficient, and interactive, and it can answer a small number of insensitive queries highly accurately – in fact, for a single query, its discrete analogue (the geometric mechanism) was recently proved to be optimal [GRS09]. However, since it handles each query independently, when used to answer k queries, the privacy cost of the Laplace mechanism scales linearly with k. Until this work this was the best known dependence on k for *interactive* privacy preserving mechanisms.

In Chapter 3 we gave a non-interactive mechanism for answering every query in some set C with privacy cost that scales with the VC dimension of C. Since the VC dimension of a set of functions is always upper bounded by the logarithm of its cardinality, this mechanism has a privacy cost that scales with $\log k$, for a set of k queries. Put another way, given a privacy budget α and a utility constraint ϵ, the mechanism of Chapter 3 can answer exponentially more queries than the Laplace mechanism. Moreover, this dependence on $\log k$ was shown to be necessary for any privacy preserving mechanism (see Chapter 5). Unfortunately, the mechansim of Chapter 3 cannot be implemented interactively, and was not known how to be implemented efficiently.

Dwork et al. [DNR$^+$09] show that modulo cryptographic assumptions, no mechanism that has a privacy cost that depends only logarithmically in k can be implemented in time poly$(\log k, \log |X|)$, even non-interactively (although the question remains open for specific query classes of interest). They also give a mechanism that runs in time poly$(k, |X|)$ which has privacy cost which scales $O(2^{\sqrt{\log k}})$. Unfortunately, their mechanism also cannot be implemented interactively.

In this paper, we give a mechanism which can be implemented interactively, and that guarantees privacy cost which scales polylogarithmically in k. This is the first interactive mechanism which has a sublinear dependence on k, and yields an exponential improve-

[5]This may take the form of, for example, a 'synthetic dataset' that is accurate on the set of queries posed to the mechanism, but is nevertheless privacy preserving. Other kinds of data structures have been proposed, however. See Chapter 3 and [DNR$^+$09]

ment in its dependence on k even compared to the best polynomial time non-interactive mechanism. We first present the algorithm in an inefficient form, and then show how to implement it in time polynomial in $|X|$ by sampling from convex polytopes.

4.2 The Median Mechanism: Basic Implementation

We now describe the median mechanism and our basic implementation of it. As described in the Introduction, the mechanism is conceptually simple. It classifies queries as "easy" or "hard", essentially according to whether or not a majority of the databases consistent with previous answers to hard queries would give an accurate answer to it (in which case the user already "knows the answer" if it is told that the query is easy). Easy queries are answered using the corresponding median value; hard queries are answered as in the Laplace mechanism.

To explain the mechanism precisely, we need to discuss a number of parameters. We take the privacy parameter α, the accuracy parameter ϵ, and the number k of queries as input; these are hard constraints on the performance of our mechanism.[6] Our mechanism obeys these constraints with a value of δ that is inverse polynomial in k and n, and a value of τ that is negligible in k and n, provided n is sufficiently large (at least polylogarithmic in k and $|X|$, see Theorem 18). Of course, such a result can be rephrased as a nearly exponential lower bound on the number of queries k that can be successfully answered as a function of the database size n.[7]

The median mechanism is shown in Figure 4.1, and it makes use of several additional parameters. For our analysis, we set their values to:

$$m = \frac{160000 \ln k \ln \frac{1}{\epsilon}}{\epsilon^2}; \tag{4.1}$$

$$\alpha' = \frac{\alpha}{720 m \ln |X|} = \Theta\left(\frac{\alpha\epsilon^2}{\log |X| \log k \log \frac{1}{\epsilon}}\right); \tag{4.2}$$

$$\gamma = \frac{4}{\alpha'\epsilon n} \ln \frac{2k}{\alpha} = \Theta\left(\frac{\log |X| \log^2 k \log \frac{1}{\epsilon}}{\alpha\epsilon^3 n}\right). \tag{4.3}$$

[6] We typically think of α, ϵ as small constants, though our results remain meaningful for some sub-constant values of α and ϵ as well. We always assume that α is at least inverse polynomial in k. Note that when α or ϵ is sufficiently small (at most c/n for a small constant c, say), simultaneously meaningful privacy and utility is clearly impossible.

[7] In contrast, the number of queries that the Laplace mechanism can privately and usefully answer is at most linear.

1. Initialize $C_0 = \{$ databases of size m over X $\}$.

2. For each query $f_1, f_2, \ldots, f_k$ in turn:

   (a) Define r_i as in (4.4) and let $\hat{r}_i = r_i + \text{Lap}(\frac{2}{\epsilon n \alpha'})$.

   (b) Let $t_i = \frac{3}{4} + j \cdot \gamma$, where $j \in \{0, 1, \ldots, \frac{1}{\gamma}\frac{3}{20}\}$ is chosen with probability proportional to 2^{-j}.

   (c) If $\hat{r}_i \geq t_i$, set a_i to be the median value of f_i on C_{i-1}.

   (d) If $\hat{r}_i < t_i$, set a_i to be $f_i(D) + \text{Lap}(\frac{1}{n\alpha'})$.

   (e) If $\hat{r}_i < t_i$, set C_i to the databases S of C_{i-1} with $|f_i(S) - a_i| \leq \epsilon/50$; otherwise $C_i = C_{i-1}$.

   (f) If $\hat{r}_j < t_j$ for more than $20m \log |X|$ values of $j \leq i$, then halt and report failure.

Figure 4.1: The Median Mechanism.

The denominator in (4.2) can be thought of as our "privacy cost" as a function of the number of queries k. Needless to say, we made no effort to optimize the constants.

The value r_i in Step 2(a) of the median mechanism is defined as

$$r_i = \frac{\sum_{S \in C_{i-1}} \exp(-\epsilon^{-1}|f_i(D) - f_i(S)|)}{|C_{i-1}|}. \tag{4.4}$$

For the Laplace perturbations in Steps 2(a) and 2(d), recall that the distribution $\text{Lap}(\sigma)$ has the cumulative distribution function

$$F(x) = 1 - F(-x) = 1 - \frac{1}{2}e^{-x/\sigma}. \tag{4.5}$$

The motivation behind the mechanism's steps is as follows. The set C_i is the set of size-m databases consistent (up to $\pm\epsilon/50$) with previous answers of the mechanism to hard queries. The focus on databases with the small size m is justified by a VC dimension argument, see Proposition 20. Steps 2(a) and 2(b) choose a random value $\hat{r}_i$ and a random threshold t_i. The value r_i in Step 2(a) is a measure of how easy the query is, with higher numbers being easier. A more obvious measure would be the fraction of databases S in C_{i-1} for which $|f_i(S) - f_i(D)| \leq \epsilon$, but this is a highly sensitive statistic (unlike r_i, see

Lemma 7). The mechanism uses the perturbed value $\hat{r}_i$ rather than r_i to privately communicate which queries are easy and which are hard. In Step 2(b), we choose the threshold t_i at random between $3/4$ and $9/10$. This randomly shifted threshold ensures that, for every database D, there is likely to be a significant gap between r_i and t_i; such gaps are useful when optimizing the privacy guarantee. Steps 2(c) and 2(d) answer easy and hard queries, respectively. Step 2(e) updates the set of databases consistent with previous answers to hard queries. We prove in Lemma 6 that Step 2(f) occurs with at most inverse polynomial probability.

Finally, we note that the median mechanism is defined as if the total number of queries k is (approximately) known in advance. This assumption can be removed by using successively doubling "guesses" of k; this increases the privacy cost by an $O(\log k)$ factor.

4.3 Analysis of Median Mechanism

This section proves the following privacy and utility guarantees for the basic implementation of the median mechanism.

Theorem 18. *For every sequence of adaptively chosen predicate queries $f_1, \ldots, f_k$ arriving online, the median mechanism is (ϵ, δ)-useful and (α, τ)-differentially private, where τ is a negligible function of k and $|X|$, and δ is an inverse polynomial function of k and n, provided the database size n satisfies*

$$n \geq \frac{30 \ln \frac{2k}{\alpha} \log_2 k}{\alpha' \epsilon} = \Theta\left(\frac{\log |X| \log^3 k \log \frac{1}{\epsilon}}{\alpha \epsilon^3}\right). \tag{4.6}$$

We prove the utility and privacy guarantees in Sections 4.3.1 and 4.3.2, respectively.[8]

4.3.1 Utility of the Median Mechanism

Here we prove a utility guarantee for the median mechanism.

Theorem 19. *The median mechanism is (ϵ, δ)-useful, where $\delta = k \exp(-\Omega(\epsilon n \alpha'))$.*

[8]If desired, in Theorem 18 we can treat n as a parameter and solve for the error ϵ. The maximum error on any query (normalized by the database size) is $O(\log k \log^{1/3} |X| / n^{1/3} \alpha^{1/3})$; the unnormalized error is a factor of n larger.

Note that under assumption (4.6), δ is inverse polynomial in k and n.

We give the proof of Theorem 19 in three pieces: with high probability, every hard query is answered accurately (Lemma 4); every easy query is answered accurately (Lemmas 3 and 5); and the algorithm does not fail (Lemma 6). The next two lemmas follow from the definition of the Laplace distribution (4.5), our choice of δ, and trivial union bounds.

Lemma 3. *With probability at least* $1 - \frac{\delta}{2}$, $|r_i - \hat{r}_i| \leq 1/100$ *for every query* i.

Lemma 4. *With probability at least* $1 - \frac{\delta}{2}$, *every answer to a hard query is* $(\epsilon/100)$*-accurate for* D.

The next lemma shows that median answers are accurate for easy queries.

Lemma 5. *If* $|r_i - \hat{r}_i| \leq 1/100$ *for every query* i, *then every answer to an easy query is* ϵ*-accurate for* D.

Proof. For a query i, let $G_{i-1} = \{S \in C_{i-1} : |f_i(D) - f_i(S)| \leq \epsilon\}$ denote the databases of C_{i-1} on which the result of query f_i is ϵ-accurate for D. Observe that if $|G_{i-1}| \geq .51 \cdot |C_{i-1}|$, then the median value of f_i on C_{i-1} is an ϵ-accurate answer for D. Thus proving the lemma reduces to showing that $\hat{r}_i \geq 3/4$ only if $|G_{i-1}| \geq .51 \cdot |C_{i-1}|$.

Consider a query i with $|G_{i-1}| < .51 \cdot |C_{i-1}|$. Using (4.4), we have

$$\begin{aligned} r_i &= \frac{\sum_{S \in C_{i-1}} \exp(-\epsilon^{-1}|f_i(D) - f_i(S)|)}{|C_{i-1}|} \\ &\leq \frac{|G_{i-1}| + e^{-1}|C_{i-1} \setminus G_{i-1}|}{|C_{i-1}|} \\ &\leq \frac{(\frac{51}{100} + \frac{49}{100e})|C_{i-1}|}{|C_{i-1}|} \\ &< \frac{74}{100}. \end{aligned}$$

Since $|r_i - \hat{r}_i| \leq 1/100$ for every query i by assumption, the proof is complete. □

Our final lemma shows that the median mechanism does not fail and hence answers every query, with high probability; this will conclude our proof of Theorem 19. We need the following preliminary proposition, which instantiates the standard uniform convergence bound with the fact that the VC dimension of every set of k predicate queries is at most $\log_2 k$ [Vap96]. Recall the definition of the parameter m from (4.1).

Proposition 20 (Uniform Convergence Bound)**.** *For every collection of k predicate queries $f_1, \ldots, f_k$ and every database D, a database S obtained by sampling points from D uniformly at random will satisfy $|f_i(D) - f_i(S)| \leq \epsilon$ for all i except with probability δ, provided*

$$|S| \geq \frac{1}{2\epsilon^2}\left(\log k + \log\frac{2}{\delta}\right).$$

In particular, there exists a database S of size m such that for all $i \in \{1, \ldots, k\}$, $|f_i(D) - f_i(S)| \leq \epsilon/400$.

In other words, the results of k predicate queries on an arbitrarily large database can be well approximated by those on a database with size only $O(\log k)$.

Lemma 6. *If $|r_i - \hat{r}_i| \leq 1/100$ for every query i and every answer to a hard query is $(\epsilon/100)$-accurate for D, then the median mechanism answers fewer than $20m \log |X|$ hard queries (and hence answers all queries before terminating).*

Proof. The plan is to track the contraction of C_i as hard queries are answered by the median mechanism. Initially we have $|C_0| \leq |X|^m$. If the median mechanism answers a hard query i, then the definition of the mechanism and our hypotheses yield

$$r_i \leq \hat{r}_i + \frac{1}{100} < t_i + \frac{1}{100} \leq \frac{91}{100}.$$

We then claim that the size of the set $C_i = \{S \in C_{i-1} : |f_i(S) - a_i| \leq \epsilon/50\}$ is at most $\frac{94}{100}|C_{i-1}|$. For if not,

$$\begin{aligned} r_i &= \frac{\sum_{S \in C_{i-1}} \exp(-\epsilon^{-1}|f_i(S) - f_i(D)|)}{|C_{i-1}|} \\ &\geq \frac{94}{100} \cdot \exp\left(-\frac{1}{50}\right) > \frac{92}{100}, \end{aligned}$$

which is a contradiction.

Iterating now shows that the number of consistent databases decreases exponentially with the number of hard queries:

$$|C_k| \leq \left(\frac{94}{100}\right)^h |X|^m \tag{4.7}$$

if h of the k queries are hard.

On the other hand, Proposition 20 guarantees the existence of a database $S^* \in C_0$ for which $|f_i(S^*) - f_i(D)| \leq \epsilon/100$ for every query f_i. Since all answers a_i produced by the median mechanism for hard queries i are $(\epsilon/100)$-accurate for D by assumption, $|f_i(S^*) - a_i| \leq |f_i(S^*) - f_i(D)| + |f_i(D) - a_i| \leq \epsilon/50$. This shows that $S^* \in C_k$ and hence $|C_k| \geq 1$. Combining this with (4.7) gives

$$h \leq \frac{m \ln |X|}{\ln(50/47)} < 20m \ln |X|,$$

as desired. □

4.3.2 Privacy of the Median Mechanism

This section establishes the following privacy guarantee for the median mechanism.

Theorem 21. *The median mechanism is (α, τ)- differentially private, where τ is a negligible function of $|X|$ and k when n is sufficiently large (as in* (4.6)*).*

We can treat the median mechanism as if it has two outputs: a vector of answers $a \in \mathbb{R}^k$, and a vector $d \in \{0,1\}^k$ such that $d_i = 0$ if i is an easy query and $d_i = 1$ if i is a hard query. A key observation in the privacy analysis is that answers to easy queries are a function only of the previous output of the mechanism, and incur no additional privacy cost beyond the release of the bit d_i. Moreover, the median mechanism is guaranteed to produce no more than $O(m \log |X|)$ answers to hard queries. Intuitively, what we need to show is that the vector d can be released after an unusually small perturbation.

Our first lemma states that the small sensitivity of predicate queries carries over, with a $2/\epsilon$ factor loss, to the r-function defined in (4.4).

Lemma 7. *The function $r_i(D) = (\sum_{S \in C} \exp(-\epsilon^{-1}|f(D) - f(S)|)/|C|$ has sensitivity $\frac{2}{\epsilon n}$ for every fixed set C of databases and predicate query f.*

Proof. Let D and D' be neighboring databases. Then

$$\begin{aligned} r_i(D) &= \frac{\sum_{S \in C} \exp(-\epsilon^{-1}|f(D) - f(S)|)}{|C_i|} \\ &\leq \frac{\sum_{S \in C_i} \exp(-\epsilon^{-1}(|f(D') - f(S)| - n^{-1}))}{|C_i|} \\ &= \exp\left(\frac{1}{\epsilon n}\right) \cdot r_i(D') \\ &\leq \left(1 + \frac{2}{\epsilon n}\right) \cdot r_i(D') \\ &\leq r_i(D') + \frac{2}{\epsilon n} \end{aligned}$$

where the first inequality follows from the fact that the (predicate) query f has sensitivity $1/n$, the second from the fact that $e^x \leq 1 + 2x$ when $x \in [0, 1]$, and the third from the fact that $r_i(D') \leq 1$. □

The next lemma identifies nice properties of "typical executions" of the median mechanism. Consider an output (d, a) of the median mechanism with a database D. From D and (d, a), we can uniquely recover the values $r_1, \ldots, r_k$ computed (via (4.4)) in Step 2(a) of the median mechanism, with r_i depending only on the first $i-1$ components of d and a. We sometimes write such a value as $r_i(D, (d, a))$, or as $r_i(D)$ if an output (d, a) has been fixed. Call a possible threshold t_i *good* for D and (d, a) if $d_i = 0$ and $r_i(D, (d, a)) \geq t_i + \gamma$, where γ is defined as in (4.3). Call a vector t of possible thresholds *good* for D and (d, a) if all but $180m \ln |X|$ of the thresholds are good for D and (d, a).

Lemma 8. *For every database D, with all but negligible ($\exp(-\Omega(\log k \log |X|/\epsilon^2))$) probability, the thresholds t generated by the median mechanism are good for its output (d, a).*

Proof. The idea is to "charge" the probability of bad thresholds to that of answering hard queries, which are strictly limited by the median mechanism. Since the median mechanism only allows $20m \ln |X|$ of the d_i's to be 1, we only need to bound the number of queries i with output $d_i = 0$ and threshold t_i satisfying $r_i < t_i + \gamma$, where r_i is the value computed by the median mechanism in Step 2(a) when it answers the query i.

Let Y_i be the indicator random variable corresponding to the (larger) event that $r_i < t_i + \gamma$. Define Z_i to be 1 if and only if, when answering the ith query, the median mechanism chooses a threshold t_i and a Laplace perturbation Δ_i such that $r_i + \Delta_i < t_i$

(i.e., the query is classified as hard). If the median mechanism fails before reaching query i, then we define $Y_i = Z_i = 0$. Set $Y = \sum_{i=1}^{k} Y_i$ and $Z = \sum_{i=1}^{k} Z_i$. We can finish the proof by showing that Y is at most $160m \ln |X|$ except with negligible probability.

Consider a query i and condition on the event that $r_i \geq \frac{9}{10}$; this event depends only on the results of previous queries. In this case, $Y_i = 1$ only if $t_i = 9/10$. But this occurs with probability $2^{-3/20\gamma}$, which using (4.3) and (4.6) is at most $1/k$.[9] Therefore, the expected contribution to Y coming from queries i with $r_i \geq \frac{9}{10}$ is at most 1. Since t_i is selected independently at random for each i, the Chernoff bound implies that the probability that such queries contribute more than $m \ln |X|$ to Y is

$$\exp(-\Omega((m \log |X|)^2)) = \exp(-\Omega((\log k)^2 (\log |X|)^2 / \epsilon^4)).$$

Now condition on the event that $r_i < \frac{9}{10}$. Let T_i denote the threshold choices that would cause Y_i to be 1, and let s_i be the smallest such; since $r_i < \frac{9}{10}$, $|T_i| \geq 2$. For every $t_i \in T_i$, $t_i > r_i - \gamma$; hence, for every $t_i \in T_i \setminus \{s_i\}$, $t_i > r_i$. Also, our distribution on the j's in Step 2(b) ensures that $\Pr[t_i \in T_i \setminus \{s_i\}] \geq \frac{1}{2} \Pr[t_i \in T_i]$. Since the Laplace distribution is symmetric around zero and the random choices Δ_i, t_i are independent, we have

$$\begin{aligned} \mathrm{E}[Z_i] &= \Pr[t_i > r_i + \Delta_i] \\ &\geq \Pr[t_i > r_i] \cdot \Pr[\Delta_i \leq 0] \\ &\geq \tfrac{1}{4} \Pr[t_i > r_i - \gamma] \\ &= \tfrac{1}{4} \mathrm{E}[Y_i]. \end{aligned} \tag{4.8}$$

The definition of the median mechanism ensures that $Z \leq 20m \ln |X|$ with probability 1. Linearity of expectation, inequality (4.8), and the Chernoff bound imply that queries with $r_i < \frac{9}{10}$ contribute at most $159m \ln |X|$ to Y with probability at least $1 - \exp(-\Omega(\log k \log |X| / \epsilon^2))$. The proof is complete. □

We can now prove Theorem 21, namely that the median mechanism preserves (α, τ) differential privacy where τ is a negligible function of $|X|$ and k.

Proof of Theorem 21: Fix a database D, queries $f_1, \ldots, f_k$, and a subset S of possible mechanism outputs. For simplicity, we assume that all perturbations are drawn from a discretized Laplace distribution, so that the median mechanism has a countable range; the continuous case can be treated using similar arguments. Then, we can think of S as a countable set of output vector pairs (d, a) with $d \in \{0,1\}^k$ and $a \in \mathbb{R}^k$. We

[9] For simplicity, we ignore the normalizing constant in the distribution over j's in Step 2(b), which is $\Theta(1)$.

write $MM(D, f) = (d, a)$ for the event that the median mechanism classifies the queries $f = (f_1, \ldots, f_k)$ according to d and outputs the numerical answers a. If the mechanism computes thresholds t while doing so, we write $MM(D, f) = (t, d, a)$. Let $G((d, a), D)$ denote the vectors that would be good thresholds for (d, a) and D. (Recall that D and (d, a) uniquely define the corresponding $r_i(D, (d, a))$'s.)

We have

$$\begin{aligned}\Pr[MM(D, f) \in S] &= \sum_{(d,a)\in S} \Pr[MM(D, f) = (d, a)] \\ &\leq \tau + \sum_{(d,a)\in S} \Pr[MM(D, f) = (t, d, a)] \\ &= \tau + \sum_{(d,a)\in S} \sum_{t\in G((d,a),D)} \Pr[MM(D, f) = (t, d, a)]\end{aligned}$$

with some t good for $(d, a), D$, and where τ is the negligible function of Lemma 8. We complete the proof by showing that, for every neighboring database D', possible output (d, a), and thresholds t good for (d, a) and D,

$$\Pr[MM(D, f) = (t, d, a)] \leq e^{\alpha} \cdot \Pr[MM(D', f) = (t, d, a)]. \tag{4.9}$$

Fix a neighboring database D', a target output (d, a), and thresholds t good for (d, a) and D. The probability that the median mechanism chooses the target thresholds t is independent of the underlying database, and so is the same on both sides of (4.9). For the rest of the proof, we condition on the event that the median mechanism uses the thresholds t (both with database D and database D').

Let $\mathcal{E}_i$ denote the event that $MM(D, f)$ classifies the first i queries in agreement with the target output (i.e., query $j \leq i$ is deemed easy if and only if $d_j = 0$) and that its first i answers are $a_1, \ldots, a_i$. Let $\mathcal{E}'_i$ denote the analogous event for $MM(D', f)$. Observe that $\mathcal{E}_k, \mathcal{E}'_k$ are the relevant events on the left- and right-hand sides of (4.9), respectively (after conditioning on t). If (d, a) is such that the median mechanism would fail after the ℓth query, then the following proof should be applied to $\mathcal{E}_\ell, \mathcal{E}'_\ell$ instead of $\mathcal{E}_k, \mathcal{E}'_k$. We next give a crude upper bound on the ratio $\Pr[\mathcal{E}_i|\mathcal{E}_{i-1}]/\Pr[\mathcal{E}'_i|\mathcal{E}'_{i-1}]$ that holds for every query (see (4.10), below), followed by a much better upper bound for queries with good thresholds.

Imagine running the median mechanism in parallel on D, D' and condition on the events $\mathcal{E}_{i-1}, \mathcal{E}'_{i-1}$. The set C_{i-1} is then the same in both runs of the mechanism, and $r_i(D), r_i(D')$ are now fixed. Let b_i (b'_i) be 0 if $MM(D, f)$ ($MM(D', f)$) classifies query i as easy and 1 otherwise. Since $r_i(D') \in [r_i(D) \pm \frac{2}{\epsilon n}]$ (Lemma 7) and a perturbation

with distribution Lap($\frac{2}{\alpha' \epsilon n}$) is added to these values before comparing to the threshold t_i (Step 2(a)),

$$\Pr[b_i = 0 \mid \mathcal{E}_{i-1}] \leq e^{\alpha'} \Pr[b'_i = 0 \mid \mathcal{E}'_{i-1}]$$

and similarly for the events where $b_i, b'_i = 1$. Suppose that the target classification is $d_i = 1$ (a hard query), and let s_i and s'_i denote the random variables $f_i(D) + \text{Lap}(\frac{1}{\alpha' n})$ and $f_i(D') + \text{Lap}(\frac{1}{\alpha' n})$, respectively. Independence of the Laplace perturbations in Steps 2(a) and 2(d) implies that

$$\Pr[\mathcal{E}_i | \mathcal{E}_{i-1}] = \Pr[b_i = 1 \mid \mathcal{E}_{i-1}] \cdot \Pr[s_i = a_i \mid \mathcal{E}_{i-1}]$$

and

$$\Pr[\mathcal{E}'_i | \mathcal{E}'_{i-1}] = \Pr[b'_i = 1 \mid \mathcal{E}'_{i-1}] \cdot \Pr[s'_i = a_i \mid \mathcal{E}'_{i-1}].$$

Since the predicate query f_i has sensitivity $1/n$, we have

$$\Pr[\mathcal{E}_i \mid \mathcal{E}_{i-1}] \leq e^{2\alpha'} \cdot \Pr[\mathcal{E}'_i \mid \mathcal{E}'_{i-1}] \tag{4.10}$$

when $d_i = 1$.

Now suppose that $d_i = 0$, and let m_i denote the median value of f_i on C_{i-1}. Then $\Pr[\mathcal{E}_i | \mathcal{E}_{i-1}]$ is either 0 (if $m_i \neq a_i$) or $\Pr[b_i = 0 \mid \mathcal{E}_{i-1}]$ (if $m_i = a_i$); similarly, $\Pr[\mathcal{E}'_i | \mathcal{E}'_{i-1}]$ is either 0 or $\Pr[b'_i = 0 \mid \mathcal{E}'_{i-1}]$. Thus the bound in (4.10) continues to hold (even with $e^{2\alpha'}$ replaced by $e^{\alpha'}$) when $d_i = 0$.

Since α' is not much smaller than the privacy target α (recall (4.2)), we cannot afford to suffer the upper bound in (4.10) for many queries. Fortunately, for queries i with good thresholds we can do much better. Consider a query i such that t_i is good for (d, a) and D and condition again on $\mathcal{E}_{i-1}, \mathcal{E}'_{i-1}$, which fixes C_{i-1} and hence $r_i(D)$. Goodness implies that $d_i = 0$, so the arguments from the previous paragraph also apply here. We can therefore assume that the median value m_i of f_i on C_{i-1} equals a_i and focus on bounding $\Pr[b_i = 0 \mid \mathcal{E}_{i-1}]$ in terms of $\Pr[b'_i = 0 \mid \mathcal{E}'_{i-1}]$. Goodness also implies that $r_i(D) \geq t_i + \gamma$ and hence $r_i(D') \geq t_i + \gamma - \frac{2}{\epsilon n} \geq t_i + \frac{\gamma}{2}$ (by Lemma 7). Recalling from (4.3) the definition of γ, we have

$$\begin{aligned} \Pr[b'_i = 0 \mid \mathcal{E}'_{i-1}] &\geq \Pr[r_i - \hat{r}_i < \tfrac{\gamma}{2}] \\ &= 1 - \tfrac{1}{2} e^{-\gamma \alpha' \epsilon n / 4} \\ &= 1 - \frac{\alpha}{4k} \end{aligned} \tag{4.11}$$

and of course, $\Pr[b_i = 0 \mid \mathcal{E}_{i-1}] \leq 1$.

Applying (4.10) to the bad queries — at most $180m \ln |X|$ of them, since t is good for (d, a) and D — and (4.11) to the rest, we can derive

$$\begin{aligned}
\Pr[\mathcal{E}_k] &= \prod_{i=1}^{k} \Pr[\mathcal{E}_i \,:\, \mathcal{E}_{i-1}] \\
&\leq \underbrace{e^{360\alpha' m \ln |X|}}_{\leq e^{\alpha/2} \text{ by (4.2)}} \cdot \underbrace{(1 - \frac{\alpha}{4k})^{-k}}_{\leq (1+\frac{\alpha}{2k})^k \leq e^{\alpha/2}} \cdot \prod_{i=1}^{k} \Pr[\mathcal{E}'_i \,:\, \mathcal{E}'_{i-1}] \\
&\leq e^{\alpha} \cdot \Pr[\mathcal{E}'_k],
\end{aligned}$$

which completes the proof of both the inequality (4.9) and the theorem. ■

4.4 The Median Mechanism: Polynomial Time Implementation

The basic implementation of the median mechanism runs in time $|X|^{\Theta(\log k \log(1/\epsilon)/\epsilon^2)}$. This section provides an efficient implementation, running in time polynomial in n, k, and $|X|$, although with a weaker usefulness guarantee.

Theorem 22. *Assume that the database size n satisfies* (4.6)*. For every sequence of adaptively chosen predicate queries $f_1, \ldots, f_k$ arriving online, the efficient implementation of the median Mechanism is (α, τ)-differentially private for a negligible function τ. Moreover, for every fixed set $f_1, \ldots, f_k$ of queries, it is (ϵ, δ)-useful for all but a negligible fraction of* fractional *databases (equivalently, probability distributions).*

Specifically, our mechanism answers exponentially many queries for all but an $O(|X|^{-m})$ fraction of probability distributions over X drawn from the unit ℓ_1 ball, and from databases drawn from such distributions. Thus our efficient implementation always guarantees privacy, but for a given set of queries $f_1, \ldots, f_k$, there might be a negligibly small fraction of fractional histograms for which our mechanism is not useful for all k queries.

We note an important nuance of our utility guarantee: even for the small fraction of fractional histograms for which the efficient median mechanism may not satisfy our usefulness guarantee, it does not output incorrect answers: it merely halts after having answered a sufficiently large number of queries using the Laplace mechanism. Therefore, even for

this small fraction of databases, the efficient median mechanism is an improvement over the Laplace mechanism: in the worst case, it simply answers every query using the Laplace mechanism before halting, and in the best case, it is able to answer many more queries.

We give a high-level overview of the proof of Theorem 22 which we then make formal. First, why isn't the median mechanism a computationally efficient mechanism? Because C_0 has super-polynomial size $|X|^m$, and computing r_i in Step 2(a), the median value in Step 2(c), and the set C_i in Step 2(e) could require time proportional to $|C_0|$. An obvious idea is to randomly sample elements of C_{i-1} to approximately compute r_i and the median value of f_i on C_{i-1}; while it is easy to control the resulting sampling error and preserve the utility and privacy guarantees of Section 4.3, it is not clear how to sample from C_{i-1} efficiently.

We show how to implement the median mechanism in polynomial time by redefining the sets C_i to be sets of probability distributions over points in X that are consistent (up to $\pm\frac{\epsilon}{50}$) with the hard queries answered up to the ith query. Each set C_i will be a convex polytope in $\mathbb{R}^{|X|}$ defined by the intersection of at most $O(m \log |X|)$ halfspaces, and hence it will be possible to sample points from C_i approximately uniformly at random in time poly$(|X|, m)$ via the grid walk of Dyer, Frieze, and Kannan [DFK91]. Lemmas 3, 4, and 5 still hold (trivially modified to accommodate sampling error). We have to reprove Lemma 6, in a somewhat weaker form: that for all but a diminishing fraction of input databases D, the median mechanism does not abort except with probability $k \exp(-\Omega(\epsilon n \alpha'))$. As for our privacy analysis of the median mechanism, it is independent of the representation of the sets C_i and the mechanisms' failure probability, and so it need not be repeated — the efficient implementation is provably private for *all* input databases and query sequences.

We now give a formal analysis of the efficient implementation.

4.4.1 Redefining the sets C_i

We redefine the sets C_i to represent databases that can contain points fractionally, as opposed to the finite set of small discrete databases. Equivalently, we can view the sets C_i as containing probability distributions over the set of points X.

We initialize C_0 to be the ℓ_1 ball of radius m in $\mathbb{R}^{|X|}$, $mB_1^{|X|}$, intersected with the non-negative orthant:

$$C_0 = \{F \in \mathbb{R}^{|X|} : F \geq 0, ||F||_1 \leq m\}.$$

Each dimension i in $\mathbb{R}^{|X|}$ corresponds to an element $x_i \in X$. Elements $F \in C_0$ can

be viewed as fractional histograms. Note that integral points in C_0 correspond exactly to databases of size at most m.

We generalize our query functions f_i to fractional histograms in the natural way:

$$f_i(F) = \frac{1}{m} \sum_{j:f_i(x_j)=1} F_j.$$

The update operation after a hard query i is answered is the same as in the basic implementation:

$$C_i \leftarrow \left\{ F \in C_{i-1} : |f_i(F) - a_i| \leq \frac{\epsilon}{50} \right\}.$$

Note that each updating operation after a hard query merely intersects C_{i-1} with the pair of halfspaces:

$$\sum_{j:f_i(x_j)=1} F_j \leq m a_i + \frac{\epsilon m}{50} \qquad \text{and} \qquad \sum_{j:f_i(x_j)=1} F_j \geq m a_i - \frac{\epsilon m}{50};$$

and so C_i is a convex polytope for each i.

Dyer, Kannan, and Frieze [DFK91] show how to δ-approximate a random sample from a convex body $K \in \mathbb{R}^{|X|}$ in time polynomial in $|X|$ and the running time of a membership oracle for K, where δ can be taken to be exponentially small (which is more than sufficient for our purposes). Their algorithm has two requirements:

1. There must be an efficient membership oracle which can in polynomial time determine whether a point $F \in \mathbb{R}^{|X|}$ lies in K.

2. K must be 'well rounded': $B_2^{|X|} \subseteq K \subseteq |X| B_2^{|X|}$, where $B_2^{|X|}$ is the unit ℓ_2 ball in $\mathbb{R}^{|X|}$.

Since C_i is given as the intersection of a set of explicit halfspaces, we have a simple membership oracle to determine whether a given point $F \in C_i$: we simply check that F lies on the appropriate side of each of the halfspaces. This takes time $\text{poly}(|X|, m)$, since the number of halfspaces defining C_i is linear in the number of answers to hard queries given before time i, which is never more than $20m \ln |X|$. Moreover, for each i we have $C_i \subseteq C_0 \subset mB_1^{|X|} \subset mB_2^{|X|} \subset |X| B_2^{|X|}$. Finally, we can safely assume that $B_2^X \subseteq C_i$ by simply considering the convex set $C_i' = C_i + B_2^X$ instead. This will not affect our results.

Therefore, we can implement the median mechanism in time $\text{poly}(|X|, k)$ by using sets C_i as defined in this section, and sampling from them using the grid walk of [DFK91].

Estimation error in computing r_i and the median value of f_i on C_{i-1} by random sampling rather than brute force is easily controlled via the Chernoff bound and can be incorporated into the proofs of Lemmas 3 and 5 in the obvious way. It remains to prove a continuous version of Lemma 6 to show that the efficient implementation of the median mechanism is (ϵ, δ)-useful on all but a negligibly small fraction of fractional histograms F.

4.4.2 Usefulness for Almost All Distributions

We now prove an analogue of Lemma 6 to establish a usefulness guarantee for the efficient version of the median mechanism.

Definition 16. *With respect to any set of k queries $f_1, \ldots, f_k$ and for any $F^* \in C_0$, define*

$$\textit{Good}_\epsilon(F^*) = \{F \in C_0 : \max_{i \in \{1,2,\ldots,k\}} |f_i(F) - f_i(F^*)| \leq \epsilon\}$$

as the set of points that agree up to an additive ϵ factor with F^* on every query f_i.

Since databases $D \subset X$ can be identified with their corresponding histogram vectors $F \in \mathbb{R}^{|X|}$, we can also write $\textit{Good}_\epsilon(D)$ when the meaning is clear from context.

For any F^*, $\text{Good}_\epsilon(F^*)$ is a convex polytope contained inside C_0. We will prove that the efficient version of the median mechanism is (ϵ, δ)-useful for a database D if

$$\frac{\text{Vol}(\text{Good}_{\epsilon/100}(D))}{\text{Vol}(C_0)} \geq \frac{1}{|X|^{2m}}. \tag{4.12}$$

We first prove that (4.12) holds for almost every fractional histogram. For this, we need a preliminary lemma.

Lemma 9. *Let $\mathcal{L}$ denote the set of integer points inside C_0. Then with respect to an arbitrary set of k queries,*

$$C_0 \subseteq \bigcup_{F \in \mathcal{L}} \textit{Good}_{\epsilon/400}(F).$$

Proof. Every rational valued point $F \in C_0$ corresponds to some (large) database $D \subset X$ by scaling F to an integer-valued histogram. Irrational points can be arbitrarily approximated by such a finite database. By Proposition 20, for every set of k predicates $f_1, \ldots, f_k$, there is a database $F^* \subset X$ with $|F^*| = m$ such that for each i, $|f_i(F^*) - f_i(F)| \leq \epsilon/400$. Recalling that the histograms corresponding to databases of size at most m are exactly the integer points in C_0, the proof is complete. □

Lemma 10. *All but an $|X|^{-m}$ fraction of* fractional *histograms F satisfy*

$$\frac{\mathrm{Vol}(\mathit{Good}_{\epsilon/200}(F))}{\mathrm{Vol}(C_0)} \geq \frac{1}{|X|^{2m}}.$$

Proof. Let

$$\mathcal{B} = \left\{ F \in \mathcal{L} : \frac{\mathrm{Vol}(\mathrm{Good}_{\epsilon/400}(F))}{\mathrm{Vol}(C_0)} \leq \frac{1}{|X|^{2m}} \right\}.$$

Consider a randomly selected fractional histogram $F^* \in C_0$. For any $F \in \mathcal{B}$ we have:

$$\Pr[F^* \in \mathrm{Good}_{\epsilon/400}(F)] = \frac{\mathrm{Vol}(\mathrm{Good}_{\epsilon/400}(F))}{\mathrm{Vol}(C_0)} < \frac{1}{|X|^{2m}}$$

Since $|\mathcal{B}| \leq |\mathcal{L}| \leq |X|^m$, by a union bound we can conclude that except with probability $\frac{1}{|X|^m}$, $F^* \notin \mathrm{Good}_{\epsilon/400}(F)$ for any $F \in \mathcal{B}$. However, by Lemma 9, $F^* \in \mathrm{Good}_{\epsilon/400}(F')$ for some $F' \in \mathcal{L}$. Therefore, except with probability $1/|X|^m$, $F' \in \mathcal{L} \setminus \mathcal{B}$. Thus, since $\mathrm{Good}_{\epsilon/400}(F') \subseteq \mathrm{Good}_{\epsilon/200}(F^*)$, except with negligible probability, we have:

$$\frac{\mathrm{Vol}(\mathrm{Good}_{\epsilon/200}(F^*))}{\mathrm{Vol}(C_0)} \geq \frac{\mathrm{Vol}(\mathrm{Good}_{\epsilon/400}(F'))}{\mathrm{Vol}(C_0)} \geq \frac{1}{|X|^{2m}}.$$

□

We are now ready to prove the analogue of Lemma 6 for the efficient implementation of the median mechanism.

Lemma 11. *For every set of k queries $f_1, \ldots, f_k$, for all but an $O(|X|^{-m})$ fraction of fractional histograms F, the efficient implementation of the median mechanism guarantees that: The mechanism answers fewer than $40m \log |X|$ hard queries, except with probability $k \exp(-\Omega(\epsilon n \alpha'))$,*

Proof. We assume that all answers to hard queries are $\epsilon/100$ accurate, and that $|r_i - \hat{r}_i| \leq \frac{1}{100}$ for every i. By Lemmas 3 and 4 — the former adapted to accommodate approximating r_i via random sampling — we are in this case except with probability $k \exp(-\Omega(\epsilon n \alpha'))$.

We analyze how the volume of C_i contracts with the number of hard queries answered. Suppose the mechanism answers a hard query at time i. Then:

$$r_i \leq \hat{r}_i + \frac{1}{100} < t_i + \frac{1}{100} \leq \frac{91}{100}.$$

Recall $C_i = \{F \in C_{i-1} : |f_i(F) - a_i| \leq \epsilon/50\}$. Suppose that $\mathrm{Vol}(C_i) \geq \frac{94}{100}\mathrm{Vol}(C_{i-1})$. Then

$$r_i = \frac{\int_{C_{i-1}} \exp(-\epsilon^{-1}|f_i(F) - f_i(D)|)dF}{\mathrm{Vol}(C_{i-1})}$$
$$\geq \frac{94}{100}\exp\left(-\frac{1}{50}\right) > \frac{92}{100},$$

a contradiction. Therefore, we have

$$|C_k| \leq \left(\frac{94}{100}\right)^h \mathrm{Vol}(C_0), \tag{4.13}$$

if h of the k queries are hard.

Since all answers to hard queries are $\epsilon/100$ accurate, it must be that $\mathrm{Good}_{\epsilon/100}(D) \in C_k$. Therefore, for an input database D that satisfies (4.12) — and this is all but an $O(|X|^{-m})$ fraction of them, by Lemma 10 — we have

$$\mathrm{Vol}(C_k) \geq \mathrm{Vol}(\mathbf{Good}_{\epsilon/100}(D)) \geq \frac{\mathrm{Vol}(C_0)}{|X|^{2m}}. \tag{4.14}$$

Combining inequalities (4.13) and (4.14) yields

$$h \leq \frac{2m\ln|X|}{\ln\frac{50}{47}} < 40m\ln|X|,$$

as claimed. □

Lemmas 4, 5, and 11 give the following utility guarantee.

Theorem 23. *For every set $f_1, \ldots, f_k$ of queries, for all but a negligible fraction of fractional histograms F, the efficient implementation of the median mechanism is (ϵ, δ)-useful with $\delta = k\exp(-\Omega(\epsilon n\alpha'))$.*

4.4.3 Usefulness for Finite Databases

Fractional histograms correspond to probability distributions over X. Lemma 10 shows that most probability distributions are 'good' for the efficient implementation of the Median Mechanism; in fact, more is true. We next show that finite databases *sampled* from randomly selected probability distributions also have good volume properties. Together,

these lemmas show that the efficient implementation of the median mechanism will be able to answer nearly exponentially many queries with high probability, in the setting in which the private database D is drawn from some 'typical' population distribution.
DatabaseSample($|D|$):

1. Select a fractional point $F \in C_0$ uniformly at random.

2. Sample and return a database D of size $|D|$ by drawing each $x \in D$ independently at random from the probability distribution over X induced by F (i.e. sample $x_i \in X$ with probability proportional to F_i).

Lemma 12. *For $|D|$ as in* (4.6) *(as required for the Median Mechanism), a database sampled by* ***DatabaseSample****($|D|$) satisfies* (4.12) *except with probability at most $O(|X|^{-m})$.*

Proof. By lemma 10, except with probability $|X|^{-m}$, the fractional histogram F selected in step 1 satisfies

$$\frac{\text{Vol}(\text{Good}_{\epsilon/200}(F))}{\text{Vol}(C_0)} \geq \frac{1}{|X|^{2m}}.$$

By lemma 20, when we sample a database D of size $|D| \geq O((\log |X| \log^3 k \log 1/\epsilon)/\epsilon^3)$ from the probability distribution induced by F, except with probability $\delta = O(k|X|^{-\log^3 k/\epsilon})$, $\text{Good}_{\epsilon/200}(F) \subset \text{Good}_{\epsilon/100}(D)$, which gives us condition (4.12). □

We would like an analogue of lemma 10 that holds for all but a diminishing fraction of *finite* databases (which correspond to lattice points within C_0) rather than fractional points in C_0, but it is not clear how uniformly randomly sampled lattice points distribute themselves with respect to the volume of C_0. If $n >> |X|$, then the lattice will be fine enough to approximate the volume of C_0, and lemma 10 will continue to hold. We now show that *small* uniformly sampled databases will also be good for the efficient version of the median mechanism. Here, small means $n = o(\sqrt{|X|})$, which allows for databases which are still polynomial in the size of X. A tighter analysis is possible, but we opt instead to give a simple argument.

Lemma 13. *For every n such that n satisfies* (4.6) *and $n = o(\sqrt{|X|})$, all but an $O(n^2/|X|)$ fraction of databases D of size $|D| = n$ satisfy condition* (4.12).

Proof. We proceed by showing that our **DatabaseSample** procedure, which we know via lemma 12 generates databases that satisfy (4.12) with high probability, is close to uniform. Note that **DatabaseSample** first selects a *probability distribution F* uniformly at random from the positive quadrant of the ℓ_1 ball, and then samples D from F.

For any particular database D^* with $|D^*| = n$ we write $\Pr_U[D = D^*]$ to denote the probability of generating D^* when we sample a database uniformly at random, and we write $\Pr_N[D = D^*]$ to denote the probability of generating D^* when we sample a database according to **DatabaseSample**. Let R denote the event that D^* contains no duplicate elements. We begin by noting by symmetry that: $\Pr_U[D = D^*|R] = \Pr_N[D = D^*|R]$ We first argue that $\Pr_U[R]$ and $\Pr_N[R]$ are both large. We immediately have that the expected number of repetitions in database D when drawn from the uniform distribution is $\binom{n}{2}/|X|$, and so $\Pr_U[\neg R] \leq \frac{n^2}{|X|}$. We now consider $\Pr_N[R]$. Since F is a uniformly random point in the positive quadrant of the ℓ_1 ball, each coordinate F_i has the marginal of a Beta distribution: $F_i \sim \beta(1, |X|-1)$. (See, for example, [Dev86] Chapter 5). Therefore, $E[F_i^2] = \frac{2}{|X|(|X|+1)}$ and so the expected number of repetitions in database D when drawn from **DatabaseSample** is $\binom{n}{2}\sum_{i=1}^{|X|} E[F_i^2] = \frac{2\binom{n}{2}}{|X|+1} \leq \frac{2n^2}{|X|}$. Therefore, $\Pr_N[\neg R] \leq \frac{2n^2}{|X|}$.

Finally, let B be the event that database D fails to satisfy (4.12). We have:

$$\begin{aligned}
\Pr_U[B] &= \Pr_U[B|R] \cdot \Pr_U[R] + \Pr_U[B|\neg R] \cdot \Pr_U[\neg R] \\
&= \Pr_N[B|R] \cdot \Pr_U[R] + \Pr_U[B|\neg R] \cdot \Pr_U[\neg R] \\
&\leq \Pr_N[B|R] \cdot \Pr_U[R] + \Pr_U[\neg R] \\
&\leq \Pr_N[B] \cdot \frac{\Pr_U[R]}{\Pr_N[R]} + \Pr_U[\neg R] \\
&\leq \frac{\Pr_N[B]}{1 - \frac{2n^2}{|X|}} + \frac{n^2}{|X|} \\
&= O(\frac{n^2}{|X|})
\end{aligned}$$

where the last equality follows from lemma 12, which states that $\Pr_N[B]$ is negligibly small. □

We observe that we can substitute either of the above lemmas for lemma 10 in the proof of lemma 11 to obtain versions of Thoerem 23:

Corollary 24. *For every set $f_1, \ldots, f_k$ of queries, for all but a negligible fraction of databases sampled by **DatabaseSample**, the efficient implementation of the median mechanism is (ϵ, δ)-useful with $\delta = k\exp(-\Omega(\epsilon n\alpha'))$.*

Corollary 25. *For every set $f_1, \ldots, f_k$ of queries, for all but an $n^2/|X|$ fraction of uniformly randomly sampled databases of size n, the efficient implementation of the median mechanism is (ϵ, δ)-useful with $\delta = k\exp(-\Omega(\epsilon n\alpha'))$.*

4.5 Conclusion

We have shown that in the setting of predicate queries, interactivity does not pose an information theoretic barrier to differentially private data release. In particular, our dependence on the number of queries k nearly matches the optimal dependence of $\log k$ achieved in the *offline* setting in Chapter 3. We remark that our dependence on other parameters is not necessarily optimal: in particular, [DNR$^+$09] achieves a better (and optimal) dependence on ϵ. We have also shown how to implement our mechanism in time poly$(|X|, k)$, although at the cost of sacrificing worst-case utility guarantees. The question of an interactive mechanism with poly$(|X|, k)$ runtime and worst-case utility guarantees remains an interesting open question. More generally, although the lower bounds of [DNR$^+$09] seem to preclude mechanisms with run-time poly$(\log |X|)$ from answering a superlinear number of generic predicate queries, the question of achieving this runtime for specific query classes of interest (offline or online) remains largely open. Recently a representation-dependent impossibility result for the class of conjunctions was obtained by Ullman and Vadhan [UV10]: either extending this to a representation-independent impossibility result, or circumventing it by giving an efficient mechanism with a novel output representation would be very interesting.

Chapter 5

Differential Privacy and the Fat-Shattering Dimension of Linear Queries

5.1 Introduction

In this chapter, we consider databases D which are real valued vectors, and the class of queries that we consider correspond to linear combinations of the entries of D. Formally, we consider databases $D \in \mathbb{R}_+^n$, and queries of the form $q \in [0,1]^n$. The answer to query q on database D is simply the dot-product of the two vectors: $q(D) = q \cdot D$. This model has previously been considered ([DN03, DMT07, DY08, HT10]), and generalizes the class of *count queries* or *predicate queries*, which has also been well studied ([DMNS06, BLR08, DNR$^+$09, RR10, UV10]), including in chapters 3 and 4 of this thesis, as well as many other settings including *coresets*, studied by [FFKN09]. In order to see how linear queries generalize predicate queries, imagine representing the database not as a collection of elements from X, but instead as a *histogram vector* of length X, with one entry for every element in X. The value of a predicate query is then the dot product between the histogram vector of the database, and the truth table of the predicate. This of course results in a database representation that is of size $|X|$, which is potentially exponentially larger than our previous database representations – but in this chapter, we are interested in information theoretic bounds, and are not concerned with computational cost.

The *fat-shattering dimension* (FSD) of a class of real-valued functions C over some domain is a generalization of the Vapnik-Chervonenkis dimension, and characterizes a

distribution-free convergence property of the mean value of each $f \in C$ to its expectation. The fat-shattering dimension of a class of functions C is known to characterize the sample complexity necessary to PAC learn C in the agnostic framework [ABDCBH97, BLW94]: that is, ignoring computation, the sample complexity that is both necessary and sufficient to learn C in the agnostic framework is polynomially related to the fat-shattering dimension of C.

Our main result is a similar information theoretic characterization of the magnitude of the noise that must be added to the answer to each query in some class C in terms of the fat-shattering dimension of C, FSD(C). We show polynomially related information theoretic upper and lower bounds on the noise that must be added to each query in C in terms of FSD(C). This generalizes the results of Chapter 3 to linear queries, and gives the first analysis of generic linear queries using some parameter other than their cardinality. This yields the first mechanism capable of answering a possibly infinite set of generic linear queries, and the first non-trivial lower bound for infinite classes of non-boolean linear queries. As a consequence, we extend results of Kasiviswanathan et al. and Chapter 3 [KLN$^+$08, BLR08] relating the sample complexity necessary for agnostic PAC learning and private agnostic PAC learning from classes of boolean valued functions to classes of real valued functions.

5.1.1 Related Work and Our Results

Dinur and Nissim studied the special case of linear queries for which both the database and the query are elements of the boolean hypercube $\{0,1\}^n$ [DN03]. Even in this special case, they showed that there cannot be any private mechanism that answers n queries with error $o(\sqrt{n})$, because an adversary could use any such mechanism to reconstruct a $1-o(1)$ fraction of the original database, a condition which they called *blatant non-privacy*. This result was strengthened by several subsequent papers [DMT07, DY08].

Dwork et al. gave the original definition of differential privacy, as well as the Laplace mechanism, which is capable of answering any k "low sensitivity" queries (including linear queries) up to error $O(k)$. A more refined analysis of the relationship between the laplace mechanism and function sensitivity was later given by [NRS07].

In Chapter 3, we considered the question of answering *predicate queries* over a database drawn from some domain X. This can be viewed as a special case of linear queries in which the queries are restricted to lie on the boolean hypercube, and the database must be integer valued: $D \in \mathbb{Z}_+^n$. There, we give a mechanism for answering every query in some class C with noise that depends linearly on the VC-dimension of the class of

queries. This is a quantity that is at most $\log |C|$ for finite classes C, and can be finite even for *infinite* classes. In Chapter 4, we gave a mechanism which achieved similar bounds in the online model, in which the mechanism does not know the set of queries that must be answered ahead of time, and instead must answer them as they arrive. We generalize the technique of the preceding chapters to apply to general linear queries. VC-dimension is no longer an appropriate measure of query complexity in this setting, but we show that a quantity known as Fat-Shattering dimension plays an analogous role.

Hardt and Talwar give matching upper and lower bounds on the noise that must be added when answering $k \leq n$ linear queries of roughly $\Theta(\frac{\sqrt{k}\log(n/k)}{\alpha})$, and [KRSU10] give similar lower bounds. In contrast, we prove bounds in terms of different parameters, and can handle arbitrarily (even infinitely) large values of k. For finite sets of k queries, our mechanism adds noise roughly $O\left(||D||_1^{2/3} \cdot \left(\frac{\log k \log n}{\alpha}\right)^{1/3}\right)$. Note that this is significantly less noise than even the lower bound of [HT10] for $k \geq \Omega(||D||_1^{4/3})$, and to achieve low *relative* error η (i.e. error $\epsilon = \eta||D||_1$), our mechanism requires only that $||D||_1$ be poly-logarithmic in k, rather than polynomial in k. For infinite classes of queries $|C|$, the $\log k$ in our bound can be replaced with the fat shattering dimension of the class C. We also show a lower bound in terms of the fat shattering dimension of the class C, which is the first non-trivial lower bound for infinite classes of non-boolean linear queries.

5.2 Preliminaries

A database is some vector $D \in \mathbb{R}_+^n$, and a query is some vector $q \in [0,1]^n$. We write that the evaluation of q on D is $q(D) = q \cdot D$. We write $||D||_1 = \sum_{i=1}^n D_i$ to denote the ℓ_1 norm of D, and note that for any query q, $q(D) \in [0, ||D||_1]$. We let C denote a (possibly infinite) class of queries. We are interested in mechanisms that are able to provide answers a_i for each $q_i \in C$ so that the maximum error, defined to be $\max_{i \in C} |q_i(D) - a_i|$ is as small as possible. Without loss of generality, we restrict our attention to mechanisms which actually output some synthetic database: mechanisms with range $\mathcal{R} = \mathbb{R}_+^n$. That is, if our mechanism outputs some synthetic database D', we take a_i to be $q_i(D')$ for each i.[1]

We formalize our notion of utility and relative utility for a randomized mechanism M:

Definition 17 (Usefulness and Relative Usefulness)**.** *A mechanism $M : \mathbb{R}_+^n \to \mathbb{R}_+^n$ is (ϵ, δ)-useful with respect to a class of queries C if with probability at least $1 - \delta$ (over the*

[1]This is without loss of generality, because given a different representation for each answer a_i to error ϵ, it is possible to compute a synthetic database D' with error at most 2ϵ using the linear program of [DNR+09].

internal coins of the mechanism), it outputs a synthetic database D' such that:

$$\sup_{q_i \in C} |q_i(D) - q_i(D')| \leq \epsilon$$

For $0 < \eta \leq 1$, M is (η, δ)-relatively useful with respect to C for databases of size s if it is $(\eta||D||_1, \delta)$-useful with respect to C for all input databases D with $||D||_1 \geq s$.

That is, useful mechanisms should have low error for each query in C. We now define differential privacy:

Definition 18 (Differential Privacy [DMNS06])**.** *A mechanism $M : \mathbb{R}_+^n \to \mathbb{R}_+^n$ is α-differentially private, if for any two databases D_1, D_2 such that $||D_1 - D_2||_1 \leq 1$, and for any $S \subseteq \mathbb{R}_+^n$:*

$$\Pr[M(D_1) \in S] \leq e^\alpha \Pr[M(D_2) \in S]$$

The standard notion of differential privacy need only hold for mechanisms defined over integer valued databases $D_1, D_2 \in \mathbb{N}^n$, which is a weaker condition. Our upper bounds will hold for the stronger notion of differential privacy, and our lower bounds for the weaker notion. A useful observation is that arbitrary (database independent) functions of differentially private mechanisms are also differentially private:

Fact 3. *If $M : \mathbb{R}_+^n \to \mathbb{R}_+^n$ is α-differentially private, and if $f : \mathbb{R}_+^n \to \mathbb{R}_+^n$ is a (possibly randomized) function, then $f(M)$ is α-differentially private.*

5.2.1 Fat Shattering Dimension

Fat-shattering-dimension is a combinatorial property describing classes of functions of the form $f : X \to [0, 1]$ for some domain X. It is a generalization of the Vapnik-Chervonenkis-dimension, which is a property only of classes of boolean valued functions of the form $f : X \to \{0, 1\}$. In this section, we generalize these concepts slightly to classes of linear queries, where we view our linear queries as linear combinations of functions $f : X \to [0, 1]$, where we let X be the set of standard basis vectors of $\mathbb{R}^n$.

Let $B = \{e_i\}_{i=1}^n$ denote the set of n standard basis vectors of $\mathbb{R}^n$ (e_i is the vector with a 1 in the i'th coordinate, and a 0 in all other coordinates). For any $S \subseteq B$ of size $|S| = d$, we say that S is γ-shattered by C if there exists a vector $r \in [0, 1]^d$ such that for every $b \in \{0, 1\}^d$, there exists a query $q_b \in C$ such that for each $e_i \in S$:

$$q_b(e_i) \begin{cases} \geq r_i + \gamma, & \text{if } b_i = 1; \\ \leq r_i - \gamma, & \text{if } b_i = 0. \end{cases}$$

Note that since the range of each query is $[0, 1]$, γ can range from 0 to $1/2$.

Definition 19 (Fat Shattering Dimension [BLW94, KS94])**.** *The γ-fat-shattering dimension of a class of linear queries C is:*

$$\textit{FSD}_\gamma(C) = \max\{d \in \mathbb{N} : C\ \gamma - \textit{shatters some } S \subseteq B \textit{ with } |S| = d\}$$

In the special case when $\gamma = r_i = 1/2$ for all i, note that the fat shattering dimension of a class of boolean valued functions is equal to its VC-dimension.

For finite classes C, we will let $k = |C|$ denote the cardinality of C. The following observation follows immediately from the definition of fat-shattering dimension:

Observation 26. *For finite classes C, $\textit{FSD}_\gamma(C) \leq \log k$ for all $\gamma > 0$, where $k = |C|$.*

5.3 Lower Bound

In this section, we show that any α-differentially private mechanism that answers every linear query in some class C must add noise at least linear in the fat-shattering dimension of C at any scale. The bound that we prove in this section is in terms of the privacy parameter α and the fat shattering dimension of the class. It differs from the upper bound proved in the next section by several important parameters, which include a $\log n$ term and a term depending on the size of the database. Beimel et al. [BKN10] have shown that the $\log n$ term in the upper bound is necessary in some contexts. The database that we construct in our lower bound is of size $O(\gamma \cdot \text{FSD}_\gamma(C))$. Therefore, in order to prove a nontrivial lower bound on the *relative* error achievable by a private mechanism, it would be necessary to remove a factor of γ from our current bound. This is possible in the context of VC-dimension, and we conjecture that it should also be possible for a bound in terms of fat-shattering dimension, and is merely a limitation of our techniques as present. The problem of proving a tight lower bound encapsulating all of the relevant parameters remains an interesting open question. We now proceed with the lower bound:

Theorem 27. *For any δ bounded away from 1 by a constant, let M be a mechanism M that is (ϵ, δ) useful with with respect to some class of linear queries C. If M preserves α-differential privacy, then*

$$\epsilon \geq \Omega\left(\sup_{0<\gamma\leq 1/2} \frac{\gamma^2 \cdot \textit{FSD}_\gamma(C)}{e^\alpha}\right)$$

We begin with some preliminaries which allow us to prove some useful lemmas:

Given some class of linear queries C and any $\gamma > 0$, let $S \subseteq B$ be a collection of basis-vectors of size $\text{FSD}_\gamma(C)$ that are γ-shattered by C, and let $r \in [0,1]^{\text{FSD}_\gamma(C)}$ be the corresponding vector as in the definition of fat-shattering dimension. We now partition S into $1/\gamma$ pieces. For each $j \in \{1, \ldots, 1/\gamma\}$, let:

$$S^j = \{e_i \in S : (j-1) \cdot \gamma < r_i \leq j \cdot \gamma\}$$

Since the sets $\{S_j\}$ partition S, By the pigeon-hole principle, there exists some j^* such that $|S^{j^*}| \geq \gamma \cdot |S| = \gamma \cdot \text{FSD}_\gamma(C)$. Let $d = |S^{j^*}|$.

We consider subsets $T \subset S^{j^*}$ of size $|T| = d/2$. For each such subset, we consider the database $D_T = \sum_{e_i \in T} e_i$. Let $b^T \in \{0,1\}^d$ be the vector guaranteed by the definition of fat shattering dimension such that:

$$b_i^T = \begin{cases} 1, & e_i \in T; \\ 0, & \text{otherwise.} \end{cases}$$

Let $q_T \in C$ be the query that corresponds to b^T as in the definition of fat shattering dimension, and let $C_{S^{j^*}} = \{q_T : T \subseteq S^{j^*}, |T| = d/2\}$.

We first show that each function q_T takes its highest value on D_T and cannot take large values on databases $D_{T'}$ for sets T' that differ significantly from T.

Lemma 14. *For all $q_T \in C_{S^{j^*}}$ and for all $T' \subseteq S^{j^*}$ with $|T'| = d/2$:*

$$q_T(D_T) - q_T(D_{T'}) \geq \frac{\gamma}{2} \cdot |T \triangle T'|$$

Proof.

$$\begin{aligned}
q_T(D_T) - q_T(D_{T'}) &= \sum_{e_i \in T} q_T(e_i) - \sum_{e_i \in T'} q_T(e_i) \\
&= \left(\sum_{e_i \in T \cap T'} q_T(e_i) - q_T(e_i) \right) + \sum_{e_i \in T \setminus T'} q_T(e_i) \\
&\quad - \sum_{e_i \in T' \setminus T} q_T(e_i) \\
&\geq \left(\sum_{e_i \in T \setminus T'} r_i + \gamma \right) - \left(\sum_{e_i \in T' \setminus T} r_i - \gamma \right) \\
&\geq 2\gamma \cdot |T \setminus T'| - \left(\max_{i \in T' \setminus T} r_i - \min_{i \in T \setminus T'} r_i \right) \cdot |T \setminus T'| \\
&\geq \gamma \cdot |T \setminus T'|
\end{aligned}$$

where the last inequality follows from the fact that $T, T' \subset S^{j^*}$ which was constructed such that:

$$\left(\max_{i \in S^{j^*}} r_i - \min_{i \in S^{j^*}} r_i\right) \leq \gamma$$

holds. Observing that $|T \triangle T'| = 2|T \setminus T'|$ completes the proof. □

With this lemma, we are ready to prove the main technical lemma for our lower bound:

Lemma 15. *For any δ bounded away from 1 by a constant, let M be an (ϵ, δ)-useful mechanism with respect to class C. Given as input $M(D_T)$, where D_T is an unknown private database for some $T \subseteq S^{j^*}$ with $|T| = d/2$, with constant probability $1 - \delta$, there is a procedure to reconstruct a new database D_{T^*} such that $|T \triangle T^*| \leq \frac{4\epsilon}{\gamma}$.*

Proof. Suppose that mechanism M is (ϵ, δ) useful with respect to C for some constant δ bounded away from 1. Then by definition, with constant probability, given input D_T, it outputs some database D' such that for all $q_i \in C$, $|q_i(D_T) - q_i(D')| \leq \epsilon$. For each $T' \subseteq S^{j^*}$ with $|T'| = d/2$ let:

$$v(T') = q_{T'}(D_{T'}) - q_{T'}(D')$$

and let $T^* = \text{argmin}_{T'} v(T')$. Therefore, we have:

$$v(T^*) \leq v(T) = q_T(D_T) - q_T(D') \leq \epsilon \tag{5.1}$$

where the last inequality follows from the usefulness of the mechanism. We also have:

$$\begin{aligned} v(T^*) &= q_{T^*}(D_{T^*}) - q_{T^*}(D') \\ &\geq q_{T^*}(D_{T^*}) - q_{T^*}(D_T) - \epsilon \\ &\geq \frac{\gamma}{2} \cdot |T \triangle T^*| - \epsilon \end{aligned}$$

where the first inequality follows from the usefulness of the mechanism, and the second inequality follows from lemma 14. Combining this with equation 5.1, we get:

$$|T \triangle T^*| \leq \frac{4\epsilon}{\gamma}$$

□

We are now ready to prove the lower bound:

Proof of Theorem. Let $T \subset S^{j^*}$ with $|T| = d/2$ be some randomly selected subset. Let $D_T = \sum_{e_i \in T} e_i$ be the corresponding database. By lemma 15, given $M(D_T, \epsilon)$, with probability $1-\delta$ there is a procedure P to reconstruct a database D_{T^*} such that $|T \triangle T^*| \leq 4\epsilon/\gamma$. Throughout the rest of the argument, we assume that this event occurs. Let $x \in T$ be an element selected from T uniformly at random, and let $y \in S \setminus T$ be an element selected from $S \setminus T$ uniformly at random. Let $T' = T \setminus x \cup \{y\}$. Observe that:

$$\Pr[x \in P(M(D_T, \epsilon))] \geq \frac{d/2 - 2\epsilon/\gamma}{d/2} = 1 - \frac{4\epsilon}{\gamma \cdot d}$$

$$\Pr[x \in P(M(D_{T'}, \epsilon))] \leq \frac{2\epsilon/\gamma}{d/2} = \frac{4\epsilon}{\gamma \cdot d}$$

Since $||D_T - D_{T'}||_1 \leq 2$, we have by the definition of α-differential privacy and fact 3:

$$\begin{aligned} e^\alpha &\geq \frac{\Pr[x \in P(M(D_T, \epsilon))]}{\Pr[x \in P(M(D_{T'}, \epsilon))]} \\ &\geq \frac{1 - \frac{4\epsilon}{\gamma \cdot d}}{\frac{4\epsilon}{\gamma \cdot d}} \\ &= \frac{\gamma \cdot d}{4\epsilon} - 1 \end{aligned}$$

Solving for ϵ, we find that:

$$\epsilon \geq \Omega\left(\frac{\gamma \cdot d}{e^\alpha}\right)$$

Since this holds for all choices of γ, the claim follows from the fact that $d \geq \gamma \text{FSD}_\gamma(C)$. □

5.4 Upper Bound

We now show that (ignoring the other important parameters), it is sufficient to add noise linear in the fat shattering dimension of C to simultaneously guarantee usefulness with respect to C and differential privacy. Unlike our lower bound which was not quite strong enough to state in terms of relative error, our upper bound is most naturally stated as a bound on relative error.

We make use of a theorem of Bartlett and Long [BL95] (improving a bound of Alon et al. [ABDCBH97]) concerning the rate of convergence of uniform Glivenko-Cantelli classes with respect to their fat-shattering dimension.

Theorem 28 ([BL95] Theorem 9)**.** *Let C be a class of functions from some domain X into $[0,1]$. Then for all distributions $\mathbb{P}$ over X and for all $\eta, \delta \geq 0$:*

$$\Pr\left[\sup_{f \in C}\left|\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \mathrm{E}_{x\sim\mathbb{P}}[f(x)]\right| \geq \eta\right] \leq \delta$$

where $\{x_i\}_{i=1}^m$ are m independent draws from $\mathbb{P}$ and

$$m = O\left(\frac{1}{\eta^2}\left(d_{\eta/5}\ln^2\frac{1}{\eta} + \ln\frac{1}{\delta}\right)\right)$$

where $d_{\eta/5} = \textbf{\textit{FSD}}_{\eta/5}(C)$.

We use this theorem to prove the following useful corollary:

Corollary 29. *Let C be a class of linear functions with coefficients in $[0,1]$ from $\mathbb{R}_+^n$ to $\mathbb{R}$. For any database $D \in \mathbb{R}_+^n$, there is a database $D' \in \mathbb{N}^n$ with*

$$||D'||_1 = O\left(\frac{d_{\eta/5}}{\eta^2}\cdot\log^2\left(\frac{1}{\eta}\right)\right)$$

such that for each $q \in C$,

$$\left|q(D) - \frac{||D||_1}{||D'||_1}q(D')\right| \leq \eta||D||_1$$

where $d_{\eta/5} = \textbf{\textit{FSD}}_{\eta/5}(C)$.

Proof. Let $B = \{e_i\}_{i=1}^n$ denote the set of n standard basis vectors over $\mathbb{R}^n$. Let $\mathbb{P}_D$ be the probability distribution over B that places probability $D_i/||D||_1$ on e_i. Note that for any $q \in C$:

$$\mathrm{E}_{e_i\sim\mathbb{P}_D}[q(e_i)] = \sum_{i=1}^{n}\frac{D_i}{||D||_1}q(e_i) = \frac{1}{||D||_1}\sum_{i=1}^{n}q(D_ie_i) = \frac{q(D)}{||D||_1}$$

Let $x_1, \ldots, x_m$ be $m = O\left(\frac{1}{\eta^2}\left(d_{\eta/5}\ln^2\frac{1}{\eta} + \ln 2\right)\right)$ independent draws from $\mathbb{P}_D$, and let $D' = \sum_{i=1}^m x_i$. Then:

$$q(D') = \sum_{i=1}^{n} q(D'_ie_i) = \sum_{i=1}^{m} q(x_i)$$

By lemma 28, we have:

$$\begin{aligned}\Pr\left[\left|\frac{q(D')}{m} - \frac{q(D)}{||D||_1}\right| \geq \eta\right] &= \Pr\left[\left|\frac{1}{m}\sum_{i=1}^{m} q(x_i) - \mathrm{E}_{e_i \sim \mathbb{P}_D}[q(e_i)]\right| \geq \eta\right] \\ &\leq \frac{1}{2}\end{aligned}$$

In particular, there exists some $D' \in \mathbb{N}^n$ with $||D'||_1 = m$ that satisfies $\left|\frac{q(D')}{||D'||_1} - \frac{q(D)}{||D||_1}\right| \leq \eta$. Multiplying through by $||D||_1$ gives the desired bound. □

Armed with Corollary 29, we may now proceed to instantiate the exponential mechanism over a sparse domain, analogously to the instantiation of the exponential mechanism in chapter 4.

Definition 20 (The Exponential Mechanism [MT07])**.** *Let $\mathcal{D}$ be some domain, and let $s : \mathbb{R}_+^n \times \mathcal{D} \to \mathbb{R}$ be some* quality score *mapping database/domain-element pairs to some real value. Let*

$$\Delta_s \geq \max_{r \in \mathcal{D}} \sup_{D_1, D_2 \in \mathbb{R}_n^+ : ||D_1 - D_2||_1 \leq 1} |s(D_1, r) - s(D_2, r)|$$

be an upper bound on the ℓ_1 sensitivity of s. The exponential mechanism defined with respect to domain $\mathcal{D}$ and score s is the probability distribution (parameterized by the private database D) which outputs each $r \in \mathcal{D}$ with probability proportional to:

$$r \sim \exp\left(\frac{s(D, r) \cdot \alpha}{2\Delta_s}\right)$$

Theorem 30 (McSherry and Talwar [MT07])**.** *The exponential mechanism preserves α-differential privacy.*

We let $m = O\left(\frac{d_{\eta/5}}{\eta^2} \cdot \log^2\left(\frac{1}{\eta}\right)\right)$, and define the domain of our instantiation of the exponential mechanism to be:

$$\mathcal{D} = \{D' \in \mathbb{N}^n : ||D'||_1 = m\}$$

We note that $|\mathcal{D}| = n^m$. Finally, we sample each $D' \in \mathcal{D}$ with probability proportional to:

$$D' \sim \exp\left(-\frac{\sup_{q \in C}\left|q(D) - \frac{||D||_1}{||D'||_1} \cdot q(D')\right| \alpha}{4}\right) \tag{5.2}$$

and output the database $D_{\text{out}} \equiv \frac{||D||_1}{||D'||_1} \cdot D'$[2]. Observe that for any two databases D_1, D_2 such that $||D_1 - D_2||_1 \leq 1$ we have:

$$\begin{aligned} \sup_{q \in C} |q(D_1) - \frac{||D_1||_1}{||D'||_1} \cdot q(D')| - \sup_{q \in C} |q(D_2) - \frac{||D_2||_1}{||D'||_1} \cdot q(D')| &\leq \\ ||D_1 - D_2||_1 + \frac{|||D_1||_1 - ||D_2||_1|}{m} &\leq \\ 1 + \frac{1}{m} \end{aligned}$$

Therefore, the distribution defined in equation 5.2 is a valid instantiation of the exponential mechanism, and by [MT07] preserves α-differential privacy. It remains to show that the above instantiation of the exponential mechanism yields a useful mechanism with low error. In particular, it gives us a *relatively useful* mechanisms with respect to classes C for databases that have size linear in the fat shattering dimension of C, or only logarithmic in $|C|$ for finite classes C. This is in contrast to the bounds of [HT10] that require databases to be of size polynomial in $|C|$ before giving relatively-useful mechanisms.

Theorem 31. *For any constant δ and any query class C, there is an (η, δ)-relatively useful mechanism that preserves α-differential privacy for any database of size at least:*

$$||D||_1 \geq \tilde{\Omega}\left(\frac{FSD_{2\eta/5}(C) \log n}{\alpha \eta^3}\right)$$

Proof of Theorem. Recall that the domain $\mathcal{D}$ of our instantiation of the exponential mechanism consists of all databases $D' \in \mathbb{N}^n$ with $||D'||_1 = m$ with $m = O\left(\frac{d_{\eta/5}}{\eta^2} \cdot \log^2\left(\frac{1}{\eta}\right)\right)\}$ In particular, by corollary 29, there exists a $D^* \in \mathcal{D}$ such that:

$$\left| q(D) - \frac{||D^*||_1}{||D'||_1} q(D^*) \right| \leq \eta ||D||_1$$

By the definition of our mechanism, such a D^* is output with probability proportional to at least:

$$D^* \sim \exp(-\frac{\eta ||D||_1 \alpha}{4})$$

[2]If $||D||_1$ is not public knowledge, it can be estimated to small constant error using the Laplace mechanism [DMNS06], losing only additive constants in the approximation parameter ϵ and privacy parameter α. This does not affect our results.

Similarly, any $D^B \in \mathcal{D}$ such that $\left|q(D) - \frac{||D^B||_1}{||D'||_1} q(D^*)\right| \geq 2\eta||D||_1$ is output with probability proportional to at most:

$$D^B \sim \exp(-\frac{\eta||D||_1\alpha}{2})$$

Let $\mathcal{D}_B$ denote the set of all such D^B. Because $|\mathcal{D}| = n^m$, we have that:

$$\frac{\Pr[D' = D^*]}{\Pr[D' \in \mathcal{D}_B]} \geq \frac{\exp(-\frac{\eta||D||_1\alpha}{4})}{n^m \cdot \exp(-\frac{\eta||D||_1\alpha}{2})} = n^{-m} \cdot \exp\left(\frac{\eta||D||_1\alpha}{2}\right)$$

Rearranging terms, we have:

$$\Pr[D' \in \mathcal{D}_B] \leq n^m \exp\left(-\frac{\eta||D||_1\alpha}{2}\right)$$

Solving, we find that this bad event occurs with probability at most δ for any database D with:

$$\begin{aligned} ||D||_1 &\geq \Omega\left(\frac{m\log n}{\eta\alpha} + \log\frac{1}{\delta}\right) \\ &= \tilde{\Omega}\left(\frac{FSD_{2\eta/5}(C)\log n}{\alpha\eta^3}\right) \end{aligned}$$

□

We remark that the above mechanism is the analogue of the general release mechanism of Chapter 3, and answers linear queries in the *offline* setting, when all queries C are known to the mechanism in advance. This is not necessary, however. In the same way as above, corollary 29 can also be used to generalize the Median Mechanism of Chapter 4, to achieve roughly the same bounds, but in the *online* setting, in which queries arrive online, and the mechanism must privately answer queries as they arrive, without knowledge of future queries. This results in the following theorem:

Theorem 32. *There exists a mechanism such that for every sequence of adaptively chosen queries $q_1, q_2, \ldots$ arriving online, chosen from some (possibly infinite) set C (unknown to the mechanism), the mechanism is (η, δ) useful with respect to C and preserves (α, τ)-differential privacy, where τ is a negligible function of n, for any database D with size at least:*

$$||D||_1 \geq \tilde{\Omega}\left(\frac{FSD_{2\eta/5}(C)\log n}{\alpha\eta^3}\right)$$

Remark: Notice that for finite classes of linear queries, we may replace the fat shattering dimension in the bounds of both theorems 31 and 32 with $\log|C|$ if we so choose.

5.5 Conclusion

In this paper, we have generalized the techniques used in Chapters 3 and 4 from the class of predicate queries to the more general class of *linear* queries. This gives the first mechanism for answering every linear query from some class C with noise that is bounded by a parameter other than the cardinality of C; in particular, we have given the first mechanism for answering all of the linear queries in certain *infinite* classes of queries beyond predicate queries. We have shown that the relevant parameter is the Fat-Shattering dimension of the class, which is a generalization of VC-dimension to non-boolean valued queries. In particular (ignoring other parameters), it is necessary and sufficient to add noise proportional to the fat shattering dimension of C. Our results show, among other things, that the sample complexity needed to privately agnostically learn real valued functions is polynomially related to the sample complexity needed to non-privately agnostically learn real valued functions.

At a high level, the same technique can be applied for any class of queries, all of the answers to which can be summarized by some 'small' object. It is then sufficient to instantiate the exponential mechanism only over this much smaller set of objects (rather than the set of all databases) to obtain a useful mechanism. In the case of linear queries, we have shown that the answers to many queries can be summarized by integer valued databases with small ℓ_1 norm. An interesting future direction is to determine what types of nonlinear (but low sensitivity) queries have similar small summarizes from which useful mechanisms can be derived.

Chapter 6

Data Release in the Local Privacy Model

6.1 Introduction

So far, we have considered a *centralized* model of data privacy, in which there exists a database administrator who has direct access to the private data. What if there is instead no trusted database administrator? Even if there is a suitable trusted party, there are many reasons not to want private data aggregated by some third party. The very existence of an aggregate database of private information raises the possibility that at some *future time*, it will come into the hands of an untrusted party, either maliciously (via data theft), or as a natural result of organizational succession. A superior model – from the perspective of the owners of private data – would be a local model, in which agents could (randomly) answer questions in a differentially private manner about their own data, without ever sharing it with anyone else. In the context of predicate queries, this seems to severely limit the expressivity of a private mechanism's interaction with the data: The mechanism can ask each user whether or not her data satisfies a given predicate, and the user may flip a coin, answering truthfully only with slightly higher probability than answering falsely. In this model, which was first considered by [KLN$^+$08], what is possible?

In this chapter, we prove that it is not possible to release even the very simple class of monotone conjunctions to subconstant error in the local privacy model (In contrast, in chapters 3 and 4, we showed that in the centralized model, conjunctions can be released to inverse polynomial error). This of course rules out the release of any class that can be used to express conjunctions, to subconstant error (for example disjunctions, decision lists, halfplanes, etc.)

We observe that this can be viewed as a barrier result of sorts: because any release

mechanism that uses the private database only to answer a polynomial number of predicate queries can be implemented in the local model, no such mechanism can be used to answer conjunctions to subconstant error. But this is the technique that we used to give efficient release mechanisms for the classes of axis aligned rectangles and large margin halfspaces in chapter 3 – and very few other *computationally efficient* techniques are known. This suggests that the problem of efficiently releasing even very simple classes of predicate queries requires the development of fundamentally new techniques.

6.2 The Local Privacy Model

The local privacy model was introduced by Kasiviswanathan et al. [KLN$^+$08] in the context of learning. The local privacy model formalizes randomized response: there is no central database of private data. Instead, each individual maintains possession of their own data element (a database of size 1), and answers questions about it only in a differentially private manner. Formally, the database $D \in X^n$ is a collection of n elements from some domain X, and each $x_i \in D$ is held by an individual.

Definition 21 ([KLN$^+$08] (Local Randomizer))**.** *An α-local randomizer $R : X \to W$ is an α-differentially private algorithm that takes a database of size $n = 1$.*

In the local privacy model, algorithms may interact with the database only through a local randomizer oracle:

Definition 22 ([KLN$^+$08] (LR Oracle))**.** *an LR oracle $LR_D(\cdot, \cdot)$ takes as input an index $i \in [n]$ and an α-local randomizer R and outputs a random value $w \in W$ chosen according to the distribution $R(x_i)$, where $x_i \in D$ is the element held by the i'th individual in the database.*

Definition 23 ([KLN$^+$08] (Local Algorithm))**.** *An algorithm is α-local if it accesses the database D via the oracle LR_D, with the following restriction: If $LR_D(i, R_1), \ldots, LR_D(i, R_k)$ are the algorithm's invocations of LR_D on index i, where each R_J is an α_j-local randomizer, then $\alpha_1 + \ldots + \alpha_k \leq \alpha$*

Because differential privacy is composable, it is easy to see that α-local algorithms are α-differentially private.

Observation 33 ([KLN$^+$08])**.** *α-local algorithms are α-differentially private.*

Kasiviswanathan et al. are concerned with the task of *learning* under the constraint of differential privacy [KLN$^+$08]. In the PAC learning setting, we view the database domain to be $X = \{0,1\}^d \times \{0,1\}$, where an element $(x, y) \in X$ consists of a bitstring x of length d, and a binary label y. Given a class C of boolean predicates over $\{0,1\}^d$, we say that the error of a predicate $h \in C$ is:

$$err(h) = \frac{1}{n} \cdot |\{(x, y) \in D : h(x) \neq y\}|$$

The goal of a learning algorithm is to select a hypothesis h in C so as to minimize $err(h)$. In the (proper) PAC setting, it is assumed that there is some hypothesis $h \in C$ with $err(h) = 0$.

Definition 24 (PAC Learning)**.** *A predicate class C is PAC learnable using class C' if there exists an algorithm A such that for all concepts $h^* \in C$ and all distributions D over $\{0,1\}^d$ and all $(\epsilon, \delta) \in (0, 1/2)$, there exists an algorithm that given $z_1, \ldots, z_n$, where each $z_i = (x_1, h^*(x_i))$ where each x_i is an i.i.d. draw from D, outputs a hypothesis $h \in C'$ satisfying:*

$$Pr[err(h) \leq \epsilon] \geq 1 - \delta$$

where both n and the running time of the algorithm are bounded by a polynomial in d, ϵ, and $\log 1/\delta$.

In the context of privacy, we may view the database as defining a (discrete) distribution over $\{0,1\}^d$, and we require further that the algorithm A satisfy α-differential privacy.

Kasiviswanathan et al. show that ignoring computation, private PAC learning in the centralized privacy model is equivalent to non-private PAC learning [KLN$^+$08]. They also show that private learning in the local privacy model is equivalent to a restricted form of PAC learning known as Statistical Query Learning, due to Kearns [Kea98]. Therefore, they obtain a lower bound for learning parity functions, originally proved by [Kea98] and generalized by Blum et al. [BFJ$^+$94] for SQ-learning.

Definition 25. *For each $S \subseteq [d]$ there exists a parity function $\chi_S(x) : \{0,1\}^d \to \{0,1\}$ defined as follows:*

$$\chi_S(x) = \left(\sum_{i \in S} x_i \right) \mod 2$$

The class of parity functions is defined to be $C_p = \{\chi_S : S \subseteq [d]\}$. Note that $|C_p| = 2^d$.

Definition 26. *Let C be a class of boolean predicates $f : \{0,1\}^n \to \{0,1\}$, and let D be a distribution over $\{0,1\}^n$. The SQ-Dimension of C with respect to D is defined to*

be SQ-DIM$(C, D) = d$, where d is the largest integer such that C contains d functions $f_1, \ldots, f_d$ with the property that for all $i \neq j$ we have:

$$|\Pr_D[f_i(x) = f_j(x)] - \Pr_D[f_i(x) \neq f_j(x)]| \leq \frac{1}{d^3}$$

Theorem 34 ([BFJ+94])**.** *Let C be a class of boolean functions over $\{0,1\}^n$ and D be a distribution such that SQ-DIM$(C, D) = d \geq 16$. Then any algorithm that makes statistical queries with tolerance $1/d^{1/3}$ must make at least $d^{1/3}/2$ queries to learn C to error less than $1/2 - 1/d^3$.*

We first observe that any subset of the parity functions $C'_p \subseteq C_p$ has SQ-Dimension $|C'_p|$ with respect to the uniform distribution:

Observation 35. *Let $C'_p \subseteq C_p$ be some subset of the parity functions, and let U be the uniform distribution over $\{0,1\}^n$. Then we have SQ-DIM$(C'_p, D) = |C'_p|$.*

This follows since every two distinct parity functions $\chi_T(x), \chi_{T'}(x)$ contain some variable x_i in their symmetric difference, and so have no correlation over the uniform distribution.

Theorem 36 ([Kea98, KLN+08])**.** *Let $C'_P \subset C_P$ be any subset of parity functions with superpolynomial cardinality: $|C'_P| \geq d^{\omega(1)}$. Then any private learning algorithm in the local privacy model cannot (ϵ, δ) learn C'_P for any ϵ bounded away from $1/2$ by a constant.*

In this chapter, we extend this lower bound to show that no private algorithm in the local model can (ϵ, δ)-usefully release conjunctions (or even small subsets of conjunctions) for any $\epsilon = o(1)$.

We first observe that absent computational constraints, the problem of learning parity functions is strictly easier than the problem of usefully releasing the class of parity functions. We may view the bitstring/label pairs (x, y) in the database as single bitstrings $x \in \{0,1\}^{d+1}$, with $x_{d+1} = y$. Let $S \subseteq [d]$ be the set that defines some parity function $\chi_S(x)$ over $\{0,1\}^d$, and consider $\chi_{S \cup \{d+1\}}(x)$, a parity function over $\{0,1\}^{d+1}$. We have:

$$\begin{aligned}
\chi_{S \cup \{d+1\}}(D) &= \frac{1}{n} \left|\{x \in D : \chi_{S \cup \{d+1\}}(x) = 1\}\right| \\
&= \frac{1}{n} |\{x \in D : \chi_S(x) \neq x_{d+1}\}| \\
&= \frac{1}{n} |\{x \in D : \chi_S(x) \neq y\}| \\
&= err(\chi_S(x))
\end{aligned}$$

If we have an (ϵ, δ)-useful mechanism with respect to a subset of parity queries, we therefore also have an (ϵ, δ) learning algorithm: we simply exhaustively search through the set of all parity queries χ_S such that $y \in S$, and select the one with the lowest value, which as we have shown, corresponds to the parity query with lowest error up to an additive ϵ.

We therefore have the following corollary:

Corollary 37. *Let $C'_P \subset C_P$ be any subset of parity functions with superpolynomial cardinality: $|C'_P| \geq d^{\omega(1)}$. There does not exist an algorithm in the local-privacy model that makes only polynomially many queries and is (ϵ, δ) useful with respect to C'_P, for any ϵ bounded away from $1/2$ by a constant.*

We will now show how corollary 37 precludes a locally private release algorithm for monotone conjunctions to subconstant error.

Definition 27. *For each $S \subseteq [d]$ there exists a monotone conjunction $c_S(x) : \{0,1\}^d \to \{0,1\}$ defined as follows:*

$$c_S(x) = \prod_{i \in S} x_i$$

The class of conjunctions is defined to be $C_c = \{c_S : S \subseteq [d]\}$. Note that $|C_c| = 2^d$.

Note that monotone conjunctions are a very simple class, contained within the class of conjunctions, disjunctions, decision lists, halfplanes, and other standard predicate classes of interest. Therefore a lower bound for the class of monotone conjunctions is a serious impediment to locally-private data release. Moreover, unlike parity functions, conjunctions are *easy* to learn in the SQ-model (and therefore in the local privacy model). Therefore, our result shows a formal separation between what is possible to *learn* in the local privacy model, and what is possible to *release*.

We first observe that for each parity function $\chi_S(x)$, we can define a real valued polynomial $f_S(x) : \mathbb{R}^d \to \mathbb{R}$ such that for every $x \in \{0,1\}^d$, $f_S(x) = \chi_S(x)$: We do this in the

natural way, using multilinear polynomial interpolation:

$$\begin{aligned} f_S(x) &= \sum_{y \in \{0,1\}^d} \chi_S(y) \cdot \left(\prod_{i:y_i=1} x_i \right) \cdot \left(\prod_{i:y_i=0} (1-x_i) \right) \\ &= \sum_{y:\sum_{i \in S} y_i \mod 2 = 1} \left(\prod_{i:y_i=1} x_i \right) \cdot \left(\prod_{i:y_i=0} (1-x_i) \right) \\ &= \sum_{i=1}^{|S|} \sum_{T \subseteq S:|T|=i} (-2)^{i-1} \cdot \left(\prod_{j \in T} x_j \right) \\ &= \sum_{i=1}^{|S|} \sum_{T \subseteq S:|T|=i} (-2)^{i-1} \cdot c_T(x) \end{aligned}$$

By restricting the above derivation to the domain $\{0,1\}^d$ and by invoking linearity, it is therefore also easy to see:

$$\chi_S(D) = \sum_{i=1}^{|S|} \sum_{T \subseteq S:|T|=i} (-2)^{i-1} \cdot c_T(D) \tag{6.1}$$

for any database D.

That is, we have shown how to express parities over sets of size $|S|$ as linear combinations of conjunctions over sets of size at most $|S|$. Suppose for point of contradiction that there existed an (ϵ, δ)-useful mechanism for monotone conjunctions for some $\epsilon = o(1)$. That is, except with probability δ, some mechanism M provides answers a_S for each $c_S \in C_c$ such that $|a_S - c_S(D)| \leq \epsilon$. Then using equation 6.1, we could compute the approximate value of each parity function $\chi_S(x)$:

$$\begin{aligned} \sum_{i=1}^{|S|} \sum_{T \subseteq S:|T|=i} (-2)^{i-1} \cdot a_T &\leq \sum_{i=1}^{|S|} \sum_{T \subseteq S:|T|=i} (-2)^{i-1} \cdot (c_T(D) + (-1)^{i-1}\epsilon) \\ &= \sum_{i=1}^{|S|} \sum_{T \subseteq S:|T|=i} (-2)^{i-1} \cdot c_T(D) + \sum_{i=1}^{|S|} \sum_{T \subseteq S:|T|=i} 2^{i-1} \cdot \epsilon \\ &= \chi_S(D) + \epsilon \cdot \left(\sum_{i=1}^{S} \binom{|S|}{i} \cdot 2^{i-1} \right) \\ &= \chi_S(D) + \left(\frac{3^{|S|}-1}{2} \right) \cdot \epsilon \end{aligned}$$

where the first inequality follows from the hypothesized usefulness of the release mechanism, and the second equality follows from equation 6.1. We can similarly derive:

$$\sum_{i=1}^{|S|} \sum_{T \subseteq S : |T| = i} (-2)^{i-1} \cdot a_T \geq \chi_S(D) - \left(\frac{3^{|S|} - 1}{2} \right) \cdot \epsilon$$

Together, we have shown that an (ϵ, δ)-useful mechanism for conjunctions with up to s variables yields an (ϵ', δ)-useful mechanism for parities of up to s variables, with $\epsilon' = O(3^s \cdot \epsilon)$.

By setting $s = O(\log \frac{1}{\epsilon})$, we can take ϵ' to be $1/4$. However, since $\epsilon = o(1)$ by assumption, we have that $\log \frac{1}{\epsilon} = \omega(1)$. Let $C'_p = \{\chi_S \in C_p : |S| \leq s\}$. We have: $|C'_p| = d^{\omega(1)}$. By corollary 37, we know that ϵ' cannot be bounded away from $1/2$ by a constant, and so we have derived a contradiction. We have therefore proven the following theorem:

Theorem 38. *No release mechanism in the local privacy model that makes only $d^{O(\log(1/\epsilon))}$ queries can be (ϵ, δ) useful with respect to monotone conjunctions with at most $1/\epsilon$ variables, for any polynomially sized database D.*

Remark: Note that this is tight: for $\epsilon = O(1)$, the mechanism which simply asks for the answer to each of the $d^{O(1/\epsilon)}$ conjunctions of size at most $O(1/\epsilon)$ asks only polynomially many queries, and achieves error ϵ per query given a database D of size $|D| = \text{poly}(d, 1/\alpha)$.

6.3 Conclusion: The Local Privacy Barrier for Efficient Release Mechanisms

Theorem 38 precludes answering (even small subsets of) monotone conjunctions to subconstant error using mechanisms that operate in the local privacy model and make only polynomially many (in d) queries, given a database that is also polynomially sized in d. Clearly, this lower bound also holds for any mechanism in the centralized privacy model that *could be* implemented in the local privacy model: in particular, any mechanism that interacts with the database by only asking a polynomial number of counting queries. Unfortunately, the only existing *efficient* release mechanisms known for answering a superpolynomial number of queries work in this manner, including the release mechanisms we gave in Chapter 3 for axis aligned rectangles, and large margin halfspaces. These techniques will not extend to even very simple classes of functions that include small monotone

conjunctions as a special case (for example, halfspaces without the large margin guarantee).

Nevertheless, one of the main open questions in differential privacy is how to construct *computationally efficient* mechanisms for answering large numbers of counting queries from natural classes (We must restrict ourselves to natural classes of queries, since results of [DNR+09] suggest that there may not be any efficient mechanism for generic counting queries). In this chapter, we have shown that such mechanisms may require fundamentally new techniques: specifically, techniques that *cannot* be implemented in the local privacy model (such as the exponential mechanism, in general), but that can be implemented efficiently. This provides both a guide for future research – we must develop these techniques – and a rule of thumb to rule out approaches that cannot possibly work: those that can be implemented in the local privacy model.

Chapter 7

Differentially Private Combinatorial Optimization

7.1 Introduction

Consider the following problems:

- Assign people using a social network to one of two servers so that most pairs of friends are assigned to the same server.
- Open a certain number of HIV treatment centers so that the average commute time for patients is small.
- Open a small number of drop-off centers for undercover agents so that each agent is able to visit some site convenient to her (each providing a list of acceptable sites).

The reader may immediately recognize that (ignoring other constraints that may arise) the above problems can be simplistically modeled as instances of well-known combinatorial optimization problems: respectively the minimum cut problem, the k-median problem, and the set cover problem. While the min cut problem is known to be efficiently solvable, the k-median problem is NP hard but admits an efficient constant factor approximation. Similarly, the set cover problem is NP hard and has an efficient logarithmic approximation algorithm. Moreover, good heuristics have been designed for these problems, and hence the above problems may be considered well-studied and solved.

However, in the above scenarios and in many others, the input data (friendship relations, medical history, agents' locations) consists of sensitive information about individu-

als. Often customer expectations, legal obligations, security concerns, or a desire to gather better quality data necessitate that confidentiality of the private input data is guaranteed. This privacy constraint (formalized below), ignored by the modeling above, is often an important design goal and hence one may prefer an private algorithm that gives slightly suboptimal solutions to a non-private optimal algorithm. Given that the most benign (or even the most beneficial) of actions possibly leaks sensitive information, how should we design algorithms for the above problems? What are the fundamental trade-offs between the utility of these algorithms and the privacy guarantees they give us?

In this chapter we initiate a systematic study of the problem of designing algorithms for combinatorial optimization problems under the constraint of *differential privacy*. Informally, the differential privacy definition requires the distribution of outcomes of the computation to not change significantly (in a precise sense) when one individual changes her input data. This is a very strong privacy guarantee: no matter what auxiliary information an adversary may have, it will not be able to learn anything significant about any individual that it couldn't have learnt were the individual not to participate in the database at all. While this guarantees privacy of an individual's sensitive data, it nevertheless allows the computation to respond when many individuals change their data, as any useful computation must do.

This work explores the challenges of producing combinatorial objects by differentially private computation. One interesting feature of these problems is the interplay between efficiency and privacy in imposing approximation constraints. The exponential mechanism [MT07] is a natural approach to attack such search problems. However, as we shall see, while it gives good utility guarantees for some problems such as k-median, it may lead to solutions very far from optimal on many others like minimum cut. A second hurdle to applying the exponential mechanism is its computational inefficiency when the set of candidate outcomes is exponentially large, which is usually the case for combinatorial optimization problems. Both these factors force us to design new algorithms—algorithms that efficiently output good solutions while guaranteeing differential privacy.

7.1.1 Our Results

We first look at the minimum cut problem. Given a graph $G = (V, E)$, the goal is to partition the vertices into two sets $(S, \overline{S})$ so as to minimize the number of edges going from one set to the other. We consider the edges of the graph as the sensitive inputs and the goal is to output a partition that does not reveal the presence or absence of individual edges. It is not hard to see that we cannot expect to output the minimum cut while guaranteeing privacy. Previous results on differentially private computation [DMNS06] show that one

can solve the decision problem well, and output the *value* of the minimum cut up to an additive error of $O(\frac{1}{\epsilon})$ while guaranteeing ϵ-differential privacy.

Our results work for the search problem where a cut itself must be output: for this case, we show that an additive error of $\Omega(\frac{\log n}{\epsilon})$ is necessary to guarantee ϵ-differential privacy. This and other lower bounds in this work are information theoretic bounds that derive from the privacy constraint and hold irrespective of computational constraints. We then give an (inefficient) differentially-private algorithm that outputs a partition of V that matches this lower bound. Finally, we present a polynomial-time variant of the algorithm which gives the same error bound and satisfies a slight relaxation of differential privacy.

For the k-median problem, we treat the clients' locations as their sensitive information. Once again, we show that any private algorithm must incur a certain additive error, and give an inefficient algorithm that matches the lower bound. We cannot however expect to design a polynomial time algorithm with the same guarantees, as the problem itself is APX-hard even in the absence of privacy constraints. Hence, we instead give an algorithm with both multiplicative and additive overheads that satisfies differential privacy.

Vertex Cover and New Challenges. We next turn our attention to covering problems such as the VERTEX COVER problem, in which we must select a small set of vertices that cover an input set of (sensitive) edges. Previous work on this problem had investigated the alternate notion of "functional" privacy, and had concluded that privately approximating even the *value* of the optimal vertex cover within a factor of $n^{1-\xi}$ is not possible under standard complexity assumptions [HKKN01]. Our use of differential privacy allows us to bypass these lower bounds and report a 2-approximation to the value of the optimal cover (with a small *additive* error) in the unweighted case.

What if we want to privately output a vertex cover instead of just its value? Interestingly, covering problems differ in one crucial aspect from the k-median and minimum cut problems: the (private) data imposes hard constraints that must be satisfied by a solution. Put differently, while private data only influences the *objective function* in the k-median problem, the data determines the *constraints* defining the set of feasible solutions in the case of the vertex cover problem. And this hard covering constraint make it impossible to output an explicit yet small vertex cover while preserving privacy: any differentially private algorithm constrained to always output an explicit vertex cover must output $n-1$ vertices, as leaving out two vertices u and v leaks the information that the edge (u, v) is certainly not present, violating differential privacy.

How can we get around this hurdle? We do so by allowing the algorithm to output an orientation for *each of the* $\binom{|V|}{2}$ *possible edges*; the orientation implicitly defines a

vertex cover for any subset of edges, without needing to reveal the subset itself. In other words, instead of explicitly outputting a feasible solution, we output a variable for each possible constraint that can be used to satisfy this constraint. If we think VC as a facility location problem, one does not need to publish the list of facilities; it is enough to tell each client which location to go to. Similarly, we tell each edge which end point to use. The privacy guarantee is that any set of edges can get together, but they will not be able to learn anything about any edge not in the collusion. In the undercover agent example above, the output enables each agent to know which drop-off location to use. At the same time, it ensures that the agent cannot from this information infer the presence or absence of other agents. The privacy guarantee we give is actually significantly stronger: an arbitrary set of agents can collude and still not be able to infer the location of any other agent from the algorithm's output.

Having resolved the representation issue, we can now turn to the computational question: *given a set of edges E, how do we find an orientation efficiently but privately?* How can we use enough information about the edge set to give us a good approximation, but not so much as to violate privacy? We give an ϵ-differentially private randomized algorithm that outputs an orientation whose expected cost is at most $(2 + 16/\epsilon)$ times the optimum. For the weighted version of the problem, the approximation guarantee weakens slightly to $16(1 + 1/\epsilon)$. In both cases, as the privacy guarantees strengthen (with decreasing ϵ), the approximation guarantees weaken proportionally—this loss is essential, as we show by a lower bound proving that the $1/\epsilon$ loss is natural and necessary, regardless of computational considerations. We give similar upper and lower bounds for SET COVER and uncapacitated facility location problems.

Differential Privacy versus Truthfulness. Our results have potential implications to non-privacy related problems as well, as we discuss in the context of the recent *Combinatorial Public Project* problem, a special case of submodular maximization introduced by Papadimitriou et al. [PSS08]. They showed that the problem can admit either truthfulness or efficient approximation, but not both simultaneously. In contrast to this negative result, we show that under differential privacy (which can be interpreted as an approximate but robust alternative to truthfulness) we can achieve the same approximation factor as the best non-truthful algorithm, plus an additive logarithmic loss.

Finally, we prove a private boosting lemma. Randomized algorithms usually come with guarantees on their performance in expectation, with the understanding that independently repeating the random experiment and picking the best among the solutions can provide similar guarantees with high probability. When working with sensitive inputs, this is not a feasible approach: while each trial may leak a very small amount of informa-

tion about an individual, publishing results of repeated trials may reveal individual data. We show how to take a private algorithm that gives bounds in expectation and convert it into one that privately gives a high probability performance guarantee. Moreover, if the original algorithm is polynomial time, then so is the modified one.

Table 7.1 summarizes the bounds we prove in this paper. For each problem, from left to right, it reports the best known non-private approximation guarantees, our best efficient ϵ-differentially private algorithms, and in each case matching upper and lower bounds for inefficient ϵ-differentially private algorithms. For a few of the efficient algorithms (marked with a †) the guarantees are only for an approximate form of differential privacy, incorporating a failure probability δ, and scaling the effective value of ϵ up by $\ln(1/\delta)$.

	Non-private	Efficient Algorithms	Information Theoretic
Vertex Cover	$2 \times \mathbf{OPT}$ [Pit85]	$(2 + 16/\epsilon) \times \mathbf{OPT}$	$\Theta(1/\epsilon) \times \mathbf{OPT}$
Wtd. Vertex Cover	$2 \times \mathbf{OPT}$ [Hoc82]	$(16 + 16/\epsilon) \times \mathbf{OPT}$	$\Theta(1/\epsilon) \times \mathbf{OPT}$
Set Cover	$\ln n \times \mathbf{OPT}$ [Joh74]	$O(\ln n + \ln m/\epsilon) \times \mathbf{OPT}$	$\Theta(\ln m/\epsilon) \times \mathbf{OPT}$
Wtd. Set Cover	$\ln n \times \mathbf{OPT}$ [Chv79]	$O(\ln n(\ln m + \ln\ln n)/\epsilon) \times \mathbf{OPT}$ †	$\Theta(\ln m/\epsilon) \times \mathbf{OPT}$
Min Cut	$\mathbf{OPT}$ [FF56]	$\mathbf{OPT} + O(\ln n/\epsilon)$ †	$\mathbf{OPT} + \Theta(\ln n/\epsilon)$
CPPP	$(1 - 1/e) \times \mathbf{OPT}$ [NWF78]	$(1 - 1/e) \times \mathbf{OPT} - O(k \ln m/\epsilon)$	$\mathbf{OPT} - \Theta(k \ln(m/k)/\epsilon)$
k-Median	$(3 + \gamma) \times \mathbf{OPT}$ [AGK+04]	$6 \times \mathbf{OPT} + O(k^2 \ln^2 n/\epsilon)$	$\mathbf{OPT} + \Theta(k \ln(n/k)/\epsilon)$

Table 7.1: Summary of Results

Secure Function Evaluation vs. Differential Privacy

Prior work on Secure Function Evaluation (SFE) tells us that in fact the minimum cut in a graph can be computed in a distributed fashion in such a way that computations *reveals nothing that cannot be learnt from the output of the computation*. While this is a strong form of a privacy guarantee, it may be unsatisfying to an individual whose private data can be inferred from the privately computed output. Indeed, it is not hard to come up with instances where an attacker with some limited auxiliary information can infer the presence or absence of specific edges from local information about the minimum cut in the graph. By relaxing the whole input privacy requirement of SFE, differential privacy is able to provide unconditional per element privacy, which SFE need not provide if the output itself discloses properties of input.

7.1.2 Related Work

The problem of private approximation was introduced by Feigenbaum et al. [FIM+06], where the constraint of *functional privacy* was used to require that two inputs with the same output value (e.g. the size of an optimal vertex cover) must produce the same value under the approximation algorithm. With this constraint, Halevi et al. [HKKN01] show that approximating the value of vertex cover to within $n^{1-\xi}$ is as hard as computing the value itself, for any constant ξ. These hardness results were extended to *search* problems by Beimel et al. [BCNW06], where the constraint is relaxed to only equate those inputs whose sets of optimal solutions are identical. These results were extended and strengthened by Beimel et al. [BHN07, BMNW07].

Nonetheless, Feigenbaum et al. [FIM+06] and others show a number of positive approximation results under versions of the functional privacy model. Halevi et al. [HKKN01] provide positive results in the function privacy setting when the algorithm is permitted to leak few bits (each equivalence class of input need not produce identical output, but must be one of at most 2^b possible outcomes). Indyk and Woodruff also give some positive results for the approximation of ℓ_2 distance and a nearest neighbor problem [IW06]. However, the privacy guarantee in functional privacy is similar to SFE and does not protect sensitive data that can be inferred from the output.

7.2 Definitions

7.2.1 The Exponential Mechanism

One particularly general tool that we will often use is the exponential mechanism of [MT07]. This construction allows differentially private computation over arbitrary domains and ranges, parametrized by a query function $q(A, r)$ mapping a pair of input data set A (a multiset over some domain) and candidate result r to a real valued "score". With q and a target privacy value ϵ, the mechanism selects an output with exponential bias in favor of high scoring outputs:

$$Pr[\mathcal{E}_q^\epsilon(A) = r] \quad \propto \quad \exp(\epsilon q(A, r)) \; . \tag{7.1}$$

If the query function q has the property that any two adjacent data sets have score within Δ of each other, for all possible outputs r, the mechanism provides $2\epsilon\Delta$-differential privacy. Typically, we would normalize q so that $\Delta = 1$. We will be using this mechanism almost exclusively over discrete ranges, where we can derive the following simple

analogue of a theorem of [MT07], that the probability of a highly suboptimal output is exponentially low:

Theorem 39. *The exponential mechanism, when used to select an output $r \in R$ gives $2\epsilon\Delta$-differential privacy, letting $R_{\mathbf{OPT}}$ be the subset of R achieving $q(A,r) = \max_r q(A,r)$, ensures that*

$$\Pr[q(A, \mathcal{E}_q^\epsilon(A)) < \max_r q(A,r) - \ln(|R|/|R_{\mathbf{OPT}}|)/\epsilon - t/\epsilon] \;\leq\; \exp(-t)\,. \quad (7.2)$$

The proof of the theorem is almost immediate: any outcome with score less than $\max_r q(A,r) - \ln(|R|/|R_{\mathbf{OPT}}|)/\epsilon - t/\epsilon$ will have normalized probability at most $\exp(-t)/|R|$; each has weight at most $\exp(\mathbf{OPT} - t)|R_{\mathbf{OPT}}|/|R|$, but is normalized by at least $|R_{\mathbf{OPT}}|\exp(\mathbf{OPT})$ from the optimal outputs. As there are at most $|R|$ such outputs their cumulative probability is at most $\exp(-t)$.

7.3 Private Min-Cut

Given a graph $G = (V, E)$ the minimum cut problem is to find a cut (S, S^c) so as to minimize $E(S, S^c)$. In absence of privacy constraints, this problem is efficiently solvable exactly. However, outputting an exact solution violates privacy, as we show in Section 7.3.1. Thus, we give an algorithm to output a cut within additive $O(\log n/\epsilon)$ edges of optimal.

The algorithm has two stages: First, given a graph G, we add edges to the graph to raise the cost of the min cut to at least $4\ln n/\epsilon$, in a differentially private manner. Second, we deploy the exponential mechanism over all cuts in the graph, using a theorem of Karger to show that for graphs with min cut at least $4\ln n/\epsilon$ the number of cuts within additive t of **OPT** increases no faster than exponentially with t. Although the exponential mechanism takes time exponential in n, we can construct a polynomial time version by considering only the polynomially many cuts within $O(\ln n/\epsilon)$ of **OPT**.

Algorithm 1 The Min-Cut Algorithm

1: **Input:** $G = (V, E)$,ϵ.
2: **Let** $H_0 \subset H_1, \ldots, \subset H_{\binom{n}{2}}$ be arbitrary strictly increasing sets of edges on V.
3: **Choose** index i with probability proportional to $\exp(-\epsilon|\mathbf{OPT}(G \cup H_i) - 8\ln n/\epsilon|)$.
4: **Choose** cut C with probability proportional to $\exp(-\epsilon Cost(G \cup H_i, C))$.
5: **Output** C.

Our result relies on a result of Karger about the number of near-minimum cuts in a graph [Kar93]

Lemma 16 ([Kar93])**.** *For any graph G with min cut C, there are at most $n^{2\alpha}$ cuts of size at most αC.*

By enlarging the size of the min cut in $G \cup H_i$ to at least $4 \ln n/\epsilon$, we ensure that the number of cuts of value $\mathbf{OPT}(G \cup H_i)+t$ is bounded by $n^2 \exp(\epsilon t/2)$. The downweighting of the exponential mechanism will be able to counteract this growth in number and ensure that we select a good cut.

Theorem 40. *For any graph G, the expected cost of ALG is at most $\mathbf{OPT} + O(\ln n/\epsilon)$.*

Proof. First, we argue that the selected index i satisfies $4 \ln n/\epsilon < \mathbf{OPT}(G \cup H_i) < \mathbf{OPT}(G) + 12 \ln n/\epsilon$ with probability at least $1 - 1/n^2$. For $\mathbf{OPT} > 8 \ln n/\epsilon$, Equation 7.2 ensures that the probability of exceeding the optimal choice (H_0) by $4 \ln n/\epsilon$ is at most $1 - 1/n^2$. Likewise, for $\mathbf{OPT} < 8 \ln n/\epsilon$, there is some optimal H_i achieving min cut size $8 \ln n/\epsilon$, and the probability we end up farther away than $4 \ln n/\epsilon$ is at most $1 - 1/n^2$.

Assuming now that $\mathbf{OPT}(G \cup H_i) > 4 \ln n/\epsilon$, Karger's lemma argues that the number c_t of cuts in $G \cup H_i$ of cost at most $\mathbf{OPT}(G \cup H_i) + t$ is at most $n^2 \exp(\epsilon t/2)$. As we are assured a cut of size $\mathbf{OPT}(G \cup H_i)$ exist, each cut of size $\mathbf{OPT}(G \cup H_i) + t$ will receive probability at most $\exp(-\epsilon t)$. Put together, the probability of a cut exceeding $\mathbf{OPT}(G \cup H_i) + b$ is at most

$$\begin{aligned} \Pr[Cost(G \cup H_i, C) > \mathbf{OPT}(G \cup H_i) + b] &\leq \sum_{t>b} \exp(-\epsilon t)(c_t - c_{t-1}) && (7.3) \\ &\leq (\exp(\epsilon) - 1) \sum_{t>b} \exp(-\epsilon t) c_t && (7.4) \\ &\leq (\exp(\epsilon) - 1) \sum_{t>b} \exp(-\epsilon t/2) n^2 && (7.5) \end{aligned}$$

The sum telescopes to $\exp(-\epsilon b/2)n^2/(\exp(\epsilon/2) - 1)$, and the denominator is within a constant factor of the leading factor of $(\exp(\epsilon) - 1)$, for $\epsilon < 1$. For $b = 8 \ln n/\epsilon$, this probability becomes at most $1/n^2$. □

Theorem 41. *The algorithm above preserves 2ϵ-differential privacy.*

Remark: Note that the first instance of the exponential mechanism in our algorithm runs efficiently (since it is selecting from only $\binom{n}{2}$ objects), but the second instance does

not. However, using Karger's algorithm we can efficiently (with high probability) generate all cuts of size at most $\mathbf{OPT} + k \ln n/\epsilon$ for any constant k. If we are willing to tolerate a polynomially small $\delta = O(1/n^k)$ to achieve (ϵ, δ)-privacy, we can select only from such a pre-generated polynomially sized set, and achieve (ϵ, δ)-differential privacy efficiently. We omit the details from this extended abstract.

7.3.1 Lower Bounds

Theorem 42. *Any ϵ-differentially private algorithm for min-cut must incur an expected additive $\Omega(\ln n/\epsilon)$ cost over* **OPT***, for any $\epsilon \in (3 \ln n/n, \frac{1}{12})$.*

Proof. Consider a $\ln n/3\epsilon$-regular graph $G = (V, E)$ on n vertices such that the minimum cuts are exactly those that isolate a single vertex, and any other cut has size at least $(2 \ln n/3\epsilon)$ (a simple probabilistic argument establishes the existence of such a G; in fact a randomly chosen $\ln n/3\epsilon$-regular graph has this property with high probability).

Let M be an ϵ-differentially private algorithm for the min-cut. Given the graph G, M outputs a partition of V. Since there are $n = |V|$ singleton cuts, there exists a vertex v such that the mechanism M run on G outputs the cut $(\{v\}, V \setminus \{v\})$ with probability at most $1/n$, i.e.

$$Pr[M(V, E) = (\{v\}, V \setminus \{v\}) \leq \frac{1}{n}.$$

Now consider the graph $G' = (V, E')$, with the edges incident on v removed from G, i.e. $E' = E \setminus \{e : v \in e\}$. Since M satisfies ϵ-differential privacy and E and E' differ in at most $\ln n/2\epsilon$ edges,

$$Pr[M(V, E') = (\{v\}, V \setminus \{v\})] \leq 1/n^{1/2}.$$

Thus with probability $(1 - \frac{1}{\sqrt{n}})$, $M(G')$ outputs a cut other than the minimum cut $(\{v\}, V \setminus \{v\})$. But all other cuts, even with these edges removed, cost at least $(\ln n/3\epsilon) - 1$. Since **OPT**is zero for G', the claim follows. □

7.4 Private k-Median

We next consider a private version of the metric k-median problem: There is a pre-specified set of points V and a metric on them, $d : V \times V \to \mathbb{R}$. There is a (private) set of demand points $D \subseteq V$. We wish to select a set of medians $F \subset V$ with $|F| = k$

to minimize the quantity $\text{Cost}(F) = \sum_{v \in D} d(v, F)$ where $d(v, F) = \min_{f \in F} d(v, f)$. Let $\Delta = \max_{u,v \in V} d(u, v)$ be the diameter of the space.

As we show in Section 7.4.1, any privacy-preserving algorithm for k-median must incur an additive loss of $\Omega(\Delta \cdot k \ln(n/k)/\epsilon)$, regardless of computational constraints. We observe that running the exponential mechanism to choose one of the $\binom{n}{k}$ subsets of medians gives an (computationally inefficient) additive guarantee.

Theorem 43. *Using the exponential mechanism to pick a set of k facilities gives an $O(\binom{n}{k} poly(n))$-time ϵ-differentially private algorithm that outputs a solution with expected cost $\mathbf{OPT} + O(k\Delta \log n/\epsilon)$.*

We next give a polynomial-time algorithm that gives a slightly worse approximation guarantee. Our algorithm is based on the local search algorithm of Arya *et al.* [AGK+04]. We start with an arbitrary set of k medians, and use the exponential mechanism to look for a (usually) improving swap. After running this local search for a suitable number of steps, we select a good solution from amongst the ones seen during the local search. The following result shows that if the current solution is far from optimal, then one can find improving swaps.

Theorem 44 (Arya et al. [AGK+04])**.** *For any set $F \subseteq V$ with $|F| = k$, there exists a set of k swaps $(x_1, y_1), \ldots, (x_k, y_k)$ such that $\sum_{i=1}^{k} (\textit{Cost}(F) - \textit{Cost}(F - \{x_i\} + \{y_i\})) \geq \textit{Cost}(F) - 5\mathbf{OPT}$.*

Corollary 45. *For any set $F \subseteq V$ with $|F| = k$, there exists some swap (x, y) such that*

$$\textit{Cost}(F) - \textit{Cost}(F - \{x_i\} + \{y_i\}) \geq \frac{\textit{Cost}(F) - 5\mathbf{OPT}}{k}.$$

Algorithm 2 The k-Median Algorithm

1: **Input:** V, Demand points $D \subseteq V$, k,ϵ.
2: **let** $F_1 \subset V$ arbitrarily with $|F_1| = k$, $\epsilon' \leftarrow \epsilon/(2\Delta(T+1))$.
3: **for** $i = 1$ to T **do**
4: Select $(x, y) \in F_i \times (V \setminus F_i)$ with probability proportional to $\exp(-\epsilon' \times \text{Cost}(F_i - \{x\} + \{y\}))$.
5: **let** $F_{i+1} \leftarrow F_i - \{x\} + \{y\}$.
6: **end for**
7: Select j from $\{1, 2, \ldots, T\}$ with probability proportional to $\exp(-\epsilon' \times \text{Cost}(F_j))$.
8: **output** F_j.

Theorem 46. *Setting $T = 6k \ln n$ and $\epsilon' = \epsilon/(2\Delta(T+1))$, the k-median algorithm provides ϵ-differential privacy and except with probability $O(1/poly(n))$ outputs a solution of cost at most $6\mathbf{OPT} + O(\Delta k^2 \log^2 n/\epsilon)$.*

Proof. We first prove the privacy. Since the Cost function has sensitivity Δ, Step 4 of the algorithm preserves $2\epsilon'\Delta$ differential privacy. Since Step 4 is run at most T times and privacy composes additively, outputting all of the T candidate solutions would give us $(2\epsilon'\Delta T)$ differential privacy. Picking out a good solution from the T candidates costs us another $2\epsilon'\Delta$, leading to the stated privacy guarantee.

We next show the approximation guarantee. By Corollary 45, so long as Cost$(F_i) \geq 6\mathbf{OPT}$, there exists a swap (x, y) that reduces the cost by at least Cost$(F_i)/6k$. As there are only n^2 possible swaps, the exponential mechanism ensures through (7.2) that we are within additive $4 \ln n/\epsilon'$ with probability at least $1 - 1/n^2$. When Cost$(F_i) \geq 6\mathbf{OPT} + 24k \ln n/\epsilon'$, with probability $1 - 1/n^2$ we have Cost$(F_{i+1}) \leq (1 - 1/6k) \times$ Cost(F_i).

This multiplicative decrease by $(1 - 1/6k)$ applies for as long as Cost$(F_i) \geq 6\mathbf{OPT} + 24k \ln n/\epsilon'$. Since Cost$(F_0) \leq n\Delta$, and $n\Delta(1 - 1/6k)^T \leq \Delta \leq 24k \ln n/\epsilon'$, there must exist an $i < T$ such that Cost$(F_i) \leq 6\mathbf{OPT} + 24k \ln n/\epsilon'$, with probability at least $(1 - T/n^2)$.

Finally, by applying the exponential mechanism again in the final stage, we select from the F_i scoring within an additive $4 \ln n/\epsilon'$ of the optimal visited F_i with probability at least $1 - 1/n^2$, again by (7.2). Plugging in the value of ϵ', we get the desired result. Increasing the constants in the additive term can drive the probability of failure to an arbitrarily small polynomial. □

7.4.1 k-Median Lower Bound

Theorem 47. *Any ϵ-differentially private algorithm for the k-median problem must incur cost $\mathbf{OPT} + \Omega(\Delta \cdot k \ln(n/k)/\epsilon)$ on some inputs.*

Proof. Consider a point set $V = [n] \times [L]$ of nL points, with $L = \ln(n/k)/10\epsilon$, and a distance function $d((i, j), (i', j')) = \Delta$ whenever $i \neq i'$ and $d((i, j), (i, j')) = 0$. Let M be a differentially private algorithm that takes a subset $D \subseteq V$ and outputs a set of k locations, for some $k < \frac{n}{4}$. Given the nature of the metric space, we assume that M outputs a k-subset of $[n]$. For a set $A \subseteq [n]$, let $D_A = A \times [L]$. Let A be a size-k subset of V chosen at random.

We claim that that $\mathrm{E}_{A,M}[|M(D_A) \cap A|] \leq \frac{k}{2}$ for any ϵ-differentially private algorithm

M. Before we prove this claim, note that it implies the expected cost of $M(D_A)$ is $\frac{k}{2} \times \Delta L$, which proves the claim since $\mathbf{OPT} = 0$.

Now to prove the claim: define $\phi := \frac{1}{k}\mathrm{E}_{A,M}[|A \cap M(D_A)|]$. We can rewrite

$$k \cdot \phi = \mathrm{E}_{A,M}[|A \cap M(D_A)|] = k \cdot \mathrm{E}_{i\in[n]}\mathrm{E}_{A\setminus\{i\},M}[\mathbf{1}_{i\in M(D_A)}] \tag{7.6}$$

Now changing A to $A' := A \setminus \{i\} + \{i'\}$ for some random i' requires altering at most $2L$ elements in $D_{A'}$, which by the differential privacy guarantee should change the probability of the output by at most $e^{2\epsilon L} = (n/k)^{1/5}$. Hence

$$\mathrm{E}_{i\in[n]}\mathrm{E}_{A',M}[\mathbf{1}_{i\in M(D_{A'})}] \geq \phi \cdot (k/n)^{1/5}. \tag{7.7}$$

But the expression on the left is just k/n, since there at at most k medians. Hence $\phi \leq (k/n)^{4/5} \leq 1/2$, which proves the claim. □

Corollary 48. *Any 1-differentially private algorithm for uniform facility location that outputs the set of chosen facilities must have approximation ratio $\Omega(\sqrt{n})$.*

Proof. We consider instances defined on the uniform metric on n points, with $d(u, v) = 1$ for all u, v, and facility opening cost $f = \frac{1}{\sqrt{n}}$. Consider a 1-differentially private mechanism M when run on a randomly chosen subset A of size $k = \sqrt{n}$. Since $\mathbf{OPT}$ is $kf = 1$ for these instances, any $o(\sqrt{n})$-approximation must select at least $\frac{k}{2}$ locations from A in expectation. By an argument analogous to the above theorem, it follows that any differentially private M must output $n/20$ of the locations in expectation. This leads to a facility opening cost of $\Omega(\sqrt{n})$. □

7.5 Vertex Cover

We now turn to the problem of (unweighted) vertex cover, in which we must select a set of vertices of minimal size so that every edge in the graph is incident to at least one vertex. In the privacy-preserving version of the problem, the private information we wish to conceal is the presence of absence of each edge.

Approximating the Vertex Cover *Size*. As mentioned earlier, the problem of approximating the vertex cover size was used as the polynomial inapproximability result of [HKKN01, BCNW06] under the constraint of *functional* privacy. Despite this, any approaches to approximating the *size* of the optimal vertex cover are immediately compatible with differential privacy. E.g., the size of a maximum matching times two is a

2-approximation, and only changes in value by at most two with the presence or absence of a single edge. Hence, this value plus Laplace$(2/\epsilon)$ noise provides ϵ-differential privacy [DMNS06]. (It is important that we use maximum rather than just maximal matchings, since the size of the latter is not uniquely determined by the graph, and the presence or absence of an edge may dramatically alter the size of the solution.)

Interestingly, for *weighted* vertex cover with maximum weight $w_{\max}$ (studied in Section 7.5.3), we have to add in Lap$(w_{\max}/\epsilon)$ noise to privately estimate the weight of the optimal solution, which can be much larger than $\mathbf{OPT}$ itself. The mechanism in Section 7.5.3 avoids this barrier by only implicitly outputting the solution and gives us a $O(1/\epsilon)$ multiplicative approximation with ϵ-differential privacy.

Private approximation algorithms for the Vertex Cover *search* problem. As noted in the introduction, any differentially private algorithm for vertex cover that outputs an explicit vertex cover (a subset of the n vertices) must output a cover of size at least $n-1$ with probability 1 on any input, an essentially useless result. This lower bound exists because differential privacy requires that any output that is possible under some input is possible under all inputs. In order to address this challenge, we change our target output slightly, and allow our algorithm to output, rather than a subset of the vertices, an orientation of the edges (including non-existent edges) so as to minimize the size of the set of destinations of existing edges. Such an orientation may be viewed as a privacy-preserving set of instructions that allows for the construction of a good vertex cover in a distributed manner: in the case of the undercover agents mentioned in the introduction, the complete set of active dropoff sites (nodes) is not revealed to the agents, but an orientation on the edges tells each agent which dropoff site to use, if she is indeed an active agent.

7.5.1 Approximating Solutions: Unweighted Vertex Cover

Many algorithms exist that provide a factor-two approximation to the unweighted vertex cover, such as selecting all endpoints of a maximal matching, or repeatedly adding a random endpoint of a random uncovered edge. Although some are randomized, none that we know of have outputs that can be made to satisfy differential privacy. One simple non-private 2-approximation to vertex cover [Pit85] repeatedly selects an uncovered edge uniformly at random, and includes a random endpoint of the edge. We can view the process, equivalently, as selecting a vertex at random with probability proportional to its uncovered degree. We will take this formulation and mix in a uniform distribution over the vertices, using a weight that will grow as the number of remaining vertices decreases. As mentioned above, our goal is to output an orientation for each edge (regardless of whether

it is actually present). The way we will do this will be to output a permutation of the vertex set; we will infer from this that an edge is oriented toward whichever of its endpoints appears first in the permutation.

Let us start from $G_1 = G$, and let G_i be the graph with $n - i + 1$ vertices remaining. We will write $d_v(G)$ for the degree of vertex v in graph G. The algorithm ALG in step i chooses from the $n - i + 1$ vertices of G_i with probability proportional to $d_v(G_i) + w_i$, for an appropriate sequence $\langle w_i \rangle$. Taking $w_i = (4/\epsilon) \times (n/(n-i+1))^{1/2}$ provides ϵ-differential privacy and a $(2 + 16/\epsilon)$ approximation factor, the proof of which will follow from the forthcoming Theorem 49 and Theorem 50.

Algorithm 3 Unweighted Vertex Cover

let $n \leftarrow |V|$, $V_1 \leftarrow V, E_1 \leftarrow E$.
for $i = 1, 2, \ldots, n$ **do**
  let $w_i \leftarrow (4/\epsilon) \times \sqrt{n/(n-i+1)}$.
  pick a vertex $v \in V_i$ with probability proportional to $d_{E_i}(v) + w_i$.
  output v. **let** $V_{i+1} \leftarrow V_i \setminus \{v\}$, $E_{i+1} \leftarrow E_i \setminus (\{v\} \times V_i)$.
end for

Theorem 49 (Privacy)**.** *ALG's differential privacy guarantee is* $\max\{1/w_1, \sum_i 2/iw_i\} \leq \epsilon$ *for the settings of* w_i *above.*

Proof. For any two sets of edges A and B, and any permutation π, let d_i be the degree of the i^{th} vertex in the permutation π and let m_i be the remaining edges, both ignoring edges incident to the first $i - 1$ vertices in π.

$$\frac{\Pr[ALG(A) = \pi]}{\Pr[ALG(B) = \pi]} = \prod_{i=1}^{n} \frac{(w_i + d_i(A))/((n-i+1)w_i + 2m_i(A))}{(w_i + d_i(B))/((n-i+1)w_i + 2m_i(B))} .$$

When A and B differ in exactly one edge, $d_i(A) = d_i(B)$ for all i except the first endpoint incident to the edge in the difference. Until this term $m_i(A)$ and $m_i(B)$ differ by exactly one, and after this term $m_i(A) = m_i(B)$. The number of nodes is always equal, of course. Letting j be the index in π of the first endpoint of the edge in difference, we can cancel all terms after j and rewrite

$$\frac{\Pr[ALG(A) = \pi]}{\Pr[ALG(B) = \pi]} = \frac{w_j + d_j(A)}{w_j + d_j(B)} \times \prod_{i \leq j} \frac{(n-i+1)w_i + 2m_i(B)}{(n-i+1)w_i + 2m_i(A)} .$$

An edge may have arrived in A, in which case $m_i(A) = m_i(B) + 1$ for all $i \leq j$, and each term in the product is at most one; moreover, $d_j(A) = d_j(B) + 1$, and hence the leading term is at most $1 + 1/w_j < \exp(1/w_1)$, which is bounded by $\exp(\epsilon/2)$.

Alternately, an edge may have departed from A, in which case the lead term is no more than one, but each term in the product exceeds one and their product must now be bounded. Note that $m_i(A) + 1 = m_i(B)$ for all relevant i, and that by ignoring all other edges we only make the product larger. Simplifying, and using $1 + x \leq \exp(x)$, we see

$$\begin{aligned}
\prod_{i \leq j} \frac{(n-i+1)w_i + 2m_i(B)}{(n-i+1)w_i + 2m_i(A)} &\leq \prod_{i \leq j} \frac{(n-i+1)w_i + 2}{(n-i+1)w_i + 0} \\
&= \prod_{i \leq j} \left(1 + \frac{2}{(n-i+1)w_i}\right) \\
&\leq \exp\left(\sum_{i \leq j} \frac{2}{(n-i+1)w_i}\right)
\end{aligned}$$

The w_i are chosen so that $\sum_i 2/(n-i+1)w_i = (\epsilon/\sqrt{n}) \sum_i 1/2\sqrt{i}$ is at most ϵ.

□

Theorem 50 (Accuracy)**.** *For all G,* $\mathrm{E}[ALG(G)] \leq (2 + 2\,avg_{i \leq n}\, w_i) \times |OPT(G)| \leq (2 + 16/\epsilon)|\mathbf{OPT}(G)|$.

Proof. Let $OPT(G)$ denote an arbitrary optimal solution to the vertex cover problem on G. The proof is inductive, on the size n of G. For G with $|OPT(G)| > n/2$, the theorem holds. For G with $|OPT(G)| \leq n/2$, the expected cost of the algorithm is the probability that the chosen vertex v is incident to an edge, plus the expected cost of $ALG(G \setminus v)$.

$$\mathrm{E}[ALG(G)] = \Pr[v \text{ incident on edge}] + \mathrm{E}_v[\mathrm{E}[ALG(G \setminus v)]] \,.$$

We will bound the second term using the inductive hypothesis. To bound the first term, the probability that v is chosen incident to an edge is at most $(2mw_n + 2m)/(nw_n + 2m)$, as there are at most $2m$ vertices incident to edges. On the other hand, the probability that we pick a vertex in $OPT(G)$ is at least $(|OPT(G)|w_n + m)/(nw_n + 2m)$. Since $|OPT(G)|$ is non-negative, we conclude that

$$\Pr[v \text{ incident on edge}] \leq (2 + 2w_n)(m/(nw_n + 2m)) \leq (2 + 2w_n) \Pr[v \in OPT(G)]$$

Since $\mathbf{1}[v \in OPT(G)] \leq |OPT(G)| - |OPT(G \setminus v)|$, and using the inductive hypothesis, we get

$$\begin{aligned}
\mathrm{E}[ALG(G)] &\leq (2 + 2w_n) \times (|OPT(G)| - \mathrm{E}_v[|OPT(G \setminus v)|]) + (2 + 2 \operatorname*{avg}_{i<n} w_i) \times \mathrm{E}_v[|OPT(G \setminus v)|] \\
&= (2 + 2w_n) \times |OPT(G)| + (2 \operatorname*{avg}_{i<n} w_i - 2w_n) \times \mathrm{E}_v[|OPT(G \setminus v)|]
\end{aligned}$$

The probability that v is from an optimal vertex cover is at least $(|OPT(G)|w_i+m)/(nw_i+2m)$, as mentioned above, and (using $(a+b)/(c+d) \geq \min\{a/c, b/d\}$) is at least $\min\{|OPT(G)|/n, 1/2\} = |OPT(G)|/n$, since $|OPT(G)| < n/2$ by assumption. Thus $\mathrm{E}[|OPT(G \setminus v)|]$ is bounded above by $(1-1/n) \times |OPT(G)|$, giving

$$\mathrm{E}[ALG(G)] \leq (2+2w_n) \times |OPT(G)| + (2 \operatorname*{avg}_{i<n} w_i - 2w_n) \times (1-1/n) \times |OPT(G)| .$$

Simplification yields the claimed results, and instantiating w_i completes the proof. □

Hallucinated Edges. Here is a slightly different way to implement the intuition behind the above algorithm: imagine adding $O(1/\epsilon)$ "hallucinated" edges to each vertex (the other endpoints of these hallucinated edges being fresh "hallucinated" vertices), and then sampling vertices without replacement proportional to these altered degrees. However, once (say) $n/2$ vertices have been sampled, output the remaining vertices in random order. This view will be useful to keep in mind for the weighted vertex cover proof. (A formal analysis of this algorithm is in Section 7.6.)

7.5.2 Vertex Cover Lower Bounds

Theorem 51. *Any algorithm for the vertex cover problem that prescribes edge-orientations with ϵ-differential privacy must have an $\Omega(1/\epsilon)$ approximation guarantee, for any $\epsilon \in (\frac{1}{n}, 1]$.*

Proof. Let $V = \{1, 2, \ldots, \lceil \frac{1}{2\epsilon} \rceil\}$, and let M be an ϵ-differentially private algorithm that takes as input a private set E of edges, and outputs an orientation $M_E : V \times V \to V$, with $M_E(u,v) \in \{u, v\}$ indicating to the edge which endpoint to use. Picking two distinct vertices $u \neq v$ uniformly at random (and equating (u,v) with (v,u)), we have by symmetry:

$$\Pr_{u,v}[M_\emptyset((u,v)) \neq u] = \tfrac{1}{2}.$$

Let $\star_u = (V, \{u\} \times (V \setminus \{u\}))$ be the star graph rooted at u. Since $\star_u$ and $\emptyset$ differ in at most $\frac{1}{2\epsilon} - 1 < \frac{1}{\epsilon}$ edges and M satisfies ϵ-differential privacy, we conclude that

$$\Pr_{u,v}[M_{\star_u}((u,v)) \neq u] \geq \tfrac{1}{2e}.$$

Thus the expected cost of M when input a uniformly random $\star_u$ is at least $\frac{1}{2e} \times \lceil \frac{1}{2\epsilon} \rceil$, while $\mathbf{OPT}(\star_u)$ is 1. We can repeat this pattern arbitrarily, picking a random star from each group of $1/\epsilon$ vertices; this results in graphs with arbitrarily large vertex covers where M incurs cost $1/\epsilon$ times the cost. □

7.5.3 Weighted Vertex Cover

In the weighted vertex cover problem, each vertex V is assigned a weight $w(v)$, and the cost of any vertex cover is the sum of the weights of the participating vertices. One can extend the unweighted 2-approximation that draws vertices at random with probability proportional to their uncovered degree to a weighted 2-approximation by drawing vertices with probability proportional to their uncovered degree divided by their weight. The differentially private analog of this algorithm essentially draws vertices with probability proportional to $1/\epsilon$ plus their degree, all divided by the weight of the vertex; the algorithm we present here is based on this idea.

Define the *score* of a vertex to be $s(v) = 1/w(v)$. Our algorithm involves hallucinating edges: to each vertex, we add in $1/\epsilon$ hallucinated edges, the other endpoints of which are imaginary vertices, whose weight is considered to be ∞ (and hence has zero score). The score of an edge $e = (u, v)$ is defined to be $s(e) = s(u) + s(v)$; hence the score of a fake edge f incident on u is $s(f) = s(u)$, since its other (imaginary) endpoint has infinite weight and zero score. We will draw edges with probability proportional to their score, and then select an endpoint to output with probability proportional to its score. In addition, once a substantial number of vertices of at least a particular weight have been output, we will output the rest of those vertices.

Assume the minimum vertex weight is 1 and the maximum is 2^J. For simplicity, we round the weight of each vertex up to a power of 2, at a potential loss of a factor of two in the approximation. Define the j^{th} weight class V_j to be the set of vertices of weight 2^j. In addition, we will assume that $|V_j| = |V_{j+1}|$ for all weight classes. In order to achieve this, we hallucinate additional fake vertices. We will never actually output a hallucinated vertex. Let N_j denote $|V_j|$.

Algorithm 4 Weighted Vertex Cover

1: **while** not all vertices have been output **do**
2: **pick** an uncovered (real or hallucinated) edge $e = (u, v)$ with probability proportional to $s(e)$.
3: **output** endpoint $u \in e$ with probability proportional to $s(u)$.
4: **while** there exists some weight class V_j such that the number of nodes of class j or higher that we've output is at least $N_j/2 = |V_j|/2$ **do**
5: **pick** the smallest such value of j
6: **output** ("dump") all remaining vertices in V_j in random order.
7: **end while**
8: **end while**

We imagine the i^{th} iteration of the outer loop of the algorithm as happening at *time* i; note that one vertex is output in Step 3, whereas multiple vertices might be output in Step 6. Let $\widetilde{n}_i$ be the sum of the scores of all real vertices not output before time i, and $\widetilde{m}_i$ be the sum of the scores of all real edges not covered before time i.

Privacy Analysis

Theorem 52. *The weighted vertex cover algorithm preserves $O(\epsilon)$ differential privacy.*

Proof. Consider some potential output π of the private vertex cover algorithm, and two weighted vertex cover instances A and B that are identical except for one edge $\mathbf{e} = (p, q)$. Let p appear before q in the permutation π; since the vertex sets are the same, if the outputs of both A and B are π, then p will be output at the same time t in both executions. Let v_t be the vertex output in Step 3 at time t in such an execution; note that either $p = v_t$, or p is output in Step 6 after v_t is output.

The probability that (conditioned on the history) a surviving vertex v is output in Step 3 of the algorithm at time i is:

$$\textstyle\sum_{\text{edges } e} \Pr[\text{pick } e] \cdot \Pr[\text{output } v \mid \text{pick } e] = \sum_{e \ni v} \frac{s(e)}{\widetilde{m}_i + \widetilde{n}_i/\epsilon} \cdot \frac{s(v)}{s(e)} = \frac{(d(v)+1/\epsilon)\cdot s(v)}{\widetilde{m}_i + \widetilde{n}_i/\epsilon}.$$

Since we compare the runs of the algorithm on A and B which differ only in edge $\mathbf{e}$, these will be identical after time t when $\mathbf{e}$ is covered, and hence

$$\frac{\Pr[M(A)=\pi]}{\Pr[M(B)=\pi]} = \frac{(d_A(v_t)+1/\epsilon)s(v_t)}{(d_B(v_t)+1/\epsilon)s(v_t)} \prod_{i \le t} \left(\frac{\widetilde{m}_i^B + \widetilde{n}_i/\epsilon}{\widetilde{m}_i^A + \widetilde{n}_i/\epsilon} \right).$$

Note that if the extra edge $\mathbf{e} \in A \setminus B$ then $d_A(v_t) \le d_B(v_t) + 1$ and $\widetilde{m}_i^B \le \widetilde{m}_i^A$, so the ratio of the probabilities is at most $1 + \epsilon < \exp(\epsilon)$. Otherwise, the leading term is less than 1 and $\widetilde{m}_i^B = \widetilde{m}_i^A + s(\mathbf{e})$, and we get

$$\frac{\Pr[M(A)=\pi]}{\Pr[M(B)=\pi]} \le \prod_{i \le t} \left(1 + \frac{s(\mathbf{e})}{\widetilde{n}_i/\epsilon} \right) \le \exp\left(s(\mathbf{e}) \cdot \epsilon \cdot \textstyle\sum_{i \le t} \frac{1}{\widetilde{n}_i} \right).$$

Let T_j be the time steps $i \le t$ where vertices in V_j are output in π. $\frac{1}{\widetilde{n}_i}$. Letting 2^{j^*} be the weight of the lighter endpoint of edge $\mathbf{e}$, we can break the sum $\sum_{i \le t} \frac{1}{\widetilde{n}_i}$ into two pieces and analyze each separately:

$$\textstyle\sum_{i \le t} \frac{1}{\widetilde{n}_i} = \sum_{j \le j^*} \sum_{i \in T_j} \frac{1}{\widetilde{n}_i} + \sum_{j > j^*} \sum_{i \in T_j} \frac{1}{\widetilde{n}_i},$$

For the first partial sum, for some $j \leq j^*$, let $\sum_{i \in T_j} \frac{1}{\widetilde{n}_i} = \frac{1}{\widetilde{n}_{i_0}} + \frac{1}{\widetilde{n}_{i_1}} + \ldots + \frac{1}{\widetilde{n}_{i_\lambda}}$ such that $i_0 > i_1 > \ldots > i_\lambda$. We claim that $\widetilde{n}_{i_0} \geq 2^{-j^*} N_{j^*}/2$. Indeed, since $\mathbf{e}$ has not yet been covered, we must have output fewer than $N_{j^*}/2$ vertices from levels j^* or higher, and hence at least $N_{j^*}/2$ remaining vertices from V_{j^*} contribute to $\widetilde{n}_{i_0}$.

In each time step in T_j, at least one vertex of score 2^{-j} is output, so we have that $\widetilde{n}_{i_\ell} \geq 2^{-j^*} N_{j^*}/2 + \ell \cdot 2^{-j}$. Hence

$$\sum_{i \in T_j} \frac{1}{\widetilde{n}_i} \leq \frac{1}{2^{-j^*} N_{j^*}/2} + \frac{1}{2^{-j^*} N_{j^*}/2 + 2^{-j}} + \ldots + \frac{1}{2^{-j^*} N_{j^*}/2 + N_j \, 2^{-j}} \, .$$

Defining $\theta = 2^{-j^*+j} \cdot N_{j^*}/2$, the expression above simplifies to

$$2^j \left(\frac{1}{\theta} + \frac{1}{\theta+1} + \ldots + \frac{1}{\theta+N_j} \right) \leq 2^j \ln \left(\frac{\theta+N_j}{\theta} \right) = 2^j \ln \left(1 + \frac{N_j}{\theta} \right).$$

Now using the assumption on the size of the weight classes, we have $N_j \leq N_{j^*} \implies N_j/\theta \leq 2^{j^*-j+1}$, and hence $\sum_{i \in T_j} \frac{1}{\widetilde{n}_i} \leq (j^* - j + 2) 2^j$, for any $j \leq j^*$. Finally,

$$\sum_{j \leq j^*} \sum_{i \in T_j} \frac{1}{\widetilde{n}_i} \leq \sum_{j \leq j^*} (j^* - j + 2) 2^j = O(2^{j^*}).$$

We now consider the other partial sum $\sum_{j > j^*} \sum_{i \in T_j} \frac{1}{\widetilde{n}_i}$. For any such value of i, we know that $\widetilde{n}_i \geq 2^{-j^*} N_{j^*}/2$. Moreover, there are at most $N_{j^*}/2$ times when we output a vertex from some weight class $j \geq j^*$ before we output all of V_{j^*}; hence there are at most $N_{j^*}/2$ terms in the sum, each of which is at most $\frac{1}{2^{-j^*} N_{j^*}/2}$, giving a bound of 2^{j^*} on the second partial sum. Putting the two together, we get that

$$\frac{\Pr[M(A) = \pi]}{\Pr[M(B) = \pi]} \leq \exp(s(\mathbf{e}) \cdot \epsilon \cdot O(2^{j^*})) = \exp(O(\epsilon)),$$

using the fact that $s(\mathbf{e}) \leq 2 \cdot 2^{-j^*}$, since the lighter endpoint of $\mathbf{e}$ had weight 2^{j^*}. □

Utility Analysis

Call a vertex v *interesting* if it is incident on a real uncovered edge when it is picked. Consider the weight class V_j: let $I_j^1 \subseteq V_j$ be the set of interesting vertices output due to Steps 3, and $I_j^2 \subseteq V_j$ be the set of interesting vertices of class j output due to Step 6. The cost incurred by the algorithm is $\sum_j 2^j (|I_j^1| + |I_j^2|)$.

Lemma 17. $\mathrm{E}[\sum_j 2^j |I_j^1|] \leq \frac{4(1+\epsilon)}{\epsilon} \mathbf{OPT}$

Proof. Every interesting vertex that our algorithm picks in Steps 3 has at least one real edge incident on it, and at most $\frac{1}{\epsilon}$ hallucinated edges. Conditioned on selecting an interesting vertex v, the selection is due to a real edge with probability at least $1/(1+\frac{1}{\epsilon})$. One can show that the (non-private) algorithm $\mathcal{A}$ that selects only real edges is a 2-approximation [Pit85]. On the other hand each vertex in I_j^1 can be coupled to a step of $\mathcal{A}$ with probability $\epsilon/(1+\epsilon)$. Since we rounded up the costs by at most a factor of two, the claim follows. □

Lemma 18. $\mathrm{E}[|I_j^2|] \leq 6\,\mathrm{E}[\sum_{j' \geq j} |I_{j'}^1|]$

Proof. Let t_j denote the time that class j is dumped. Recall that by (7.5.3), we pick a surviving vertex v with probability $\propto (d(v)+\frac{1}{\epsilon}) \cdot s(v)$ at each step. This expression summed over all uninteresting vertices is $\cup_{j' \geq j} V_{j'}$ is at most $(1/\epsilon)\sum_{j' \geq j} 2^{-j'} N_{j'} \leq 2^{-j+1} N_j/\epsilon$. On the other hand, at each step before time t_j, all the interesting vertices in I_j^2 are available and the same expression summed over them is at least $2^{-j}|I_j^2|/\epsilon$. Thus for any $t \leq t_j$, conditioned on outputting a vertex $v_t \in \cup_{j' \geq j} V_{j'}$ in Step 3, the probability that it is interesting is at least $\frac{|I_j^2|2^{-j}/\epsilon}{(|I_j^2|2^{-j}+2^{1-j}N_j)/\epsilon} \geq \frac{|I_j^2|}{3N_j}$ (using $|I_j^2| \leq N_j$). Now since we output $N_j/2$ vertices from $\cup_{j' \geq j} V_{j'}$ in Step 3 before time t_j, we conclude that $\mathrm{E}\big[\sum_{j' \geq j} |I_{j'}^1| \;\big|\; |I_j^2|\big] \geq \frac{N_j}{2} \times \frac{|I_j^2|}{3N_j} = \frac{|I_j^2|}{6}$. Taking expectations completes the proof. □

We can now compute the total cost of all the interesting vertices dumped in Steps 6 of the algorithm.

$$\begin{aligned}
\mathrm{E}[\mathsf{cost}(\bigcup_j I_j^2)] &= \sum_j 2^j\,\mathrm{E}[|I_j^2|] \\
&\leq 6\sum_j 2^j \sum_{j' \geq j} \mathrm{E}[|I_{j'}^1|] \\
&\leq 6\sum_{j'} \mathrm{E}[|I_{j'}^1|]2^{j'+1} \\
&\leq 12\cdot \mathrm{E}[\mathsf{cost}(\bigcup_j I_j^1)].
\end{aligned}$$

Finally, combining this calculation with Lemma 17, we conclude that our algorithm gives an $O(\frac{1}{\epsilon})$ approximation to the weighted vertex cover problem.

7.6 Unweighted Vertex Cover Algorithm: An Alternate View

In this section, we consider a slightly different way to implement the vertex cover algorithm. Given a graph $G = (V, E)$, we mimic the randomized proportional-to-degree algorithm for αn rounds ($\alpha < 1$), and output the remaining vertices in random order. That is, in each of the first αn rounds, we select the next vertex i with probability proportional to $d(i) + 1/\epsilon$: this is equivalent to imagining that each vertex has $1/\epsilon$ "hallucinated" edges in addition to its real edges. (It is most convenient to imagine the other endpoint of these hallucinated edges as being fake vertices which are always ignored by the algorithm.)

When we select a vertex, we remove it from the graph, together with the real and hallucinated edges adjacent to it. This is equivalent to picking a random (real or hallucinated) edge from the graph, and outputting a random real endpoint. Outputting a vertex affects the real edges in the remaining graph, but does not change the hallucinated edges incident to other vertices.

Privacy Analysis. The privacy analysis is similar to that of Theorem 49: imagine the weights being $w_i = 1/\epsilon$ for the first αn rounds and $w_i = \infty$ for the remaining rounds, which gives us $2\sum_{i=(1-\alpha)n}^{n} \frac{1}{iw_i} \leq \epsilon\ (\frac{2\alpha}{1-\alpha})$-differential privacy.

Utility Analysis. To analyze the utility, we couple our algorithm with a run of the non-private algorithm $\mathcal{A}$ that at each step picks an arbitrary edge of the graph and then picks a random endpoint: it is an easy exercise that this an 2-approximation algorithm.

We refer to vertices that have non-zero "real" degree at the time they are selected by our algorithm as *interesting vertices*: the cost of our algorithm is simply the number of interesting vertices it selects in the course of its run. Let I_1 denote the number of interesting vertices it selects during the first αn steps, and I_2 denote the number of interesting vertices it selects during its remaining $(1-\alpha)n$ steps, when it is simply ordering vertices randomly. Clearly, the total cost is $I_1 + I_2$.

We may view the first phase of our algorithm as selecting an edge at random (from among both real and hallucinated ones) and then outputting one of its endpoints at random. Now, for the rounds in which our algorithm selects a real edge, we can couple this selection with one step of an imagined run of $\mathcal{A}$ (selecting the same edge and endpoint). Note that this run of $\mathcal{A}$ maintains a vertex cover that is a subset of our vertex cover, and that once our algorithm has completed a vertex cover, no interesting vertices remain. Therefore, while

our algorithm continues to incur cost, $\mathcal{A}$ has not yet found a vertex cover.

In the first phase of our algorithm, every interesting vertex our algorithm selects has at least one real edge adjacent to it, as well as $1/\epsilon$ hallucinated edges. Conditioned on selecting an interesting vertex, our algorithm had selected a real edge with probability at least $\epsilon' = 1/(1 + 1/\epsilon)$. Let R denote the random variable that represents the number of steps $\mathcal{A}$ is run for. $E[R] \leq 2\mathbf{OPT}$ since $\mathcal{A}$ is a 2-approximation algorithm. By linearity of expectation:

$$2\mathbf{OPT} \geq \mathrm{E}[R] \geq \epsilon' \cdot E[I_1] \tag{7.8}$$

We now show that most of our algorithm's cost comes from the first phase, and hence that I_2 is not much larger than I_1.

Lemma 19.

$$\mathrm{E}[I_1] \geq \ln\left(\frac{1}{1-\alpha}\right) \cdot \mathrm{E}[I_2]$$

Proof. Consider each of the αn steps of the first phase of our algorithm. Let n_i denote the number of interesting vertices remaining at step i. Note that $\{n_i\}$ is a non-increasing sequence. At step i, there are n_i interesting vertices and $n - i + 1$ remaining vertices. Note that the probability of picking an interesting vertex is strictly greater than $n_i/(n - i + 1)$ at each step. We may therefore bound the expected number of interesting vertices picked in the first phase:

$$\mathrm{E}[I_1] > \sum_{i=1}^{\alpha n} \frac{\mathrm{E}[n_i]}{n-i+1} \geq \mathrm{E}[n_{\alpha n}] \sum_{i=(1-\alpha)n}^{n} \frac{1}{i} \geq \ln\left(\frac{1}{1-\alpha}\right) \cdot \mathrm{E}[n_{\alpha n}]$$

Noting that $\mathrm{E}[I_2] \leq \mathrm{E}[n_{\alpha n}]$ completes the proof. □

Combining the facts above, we get that

$$\frac{\mathrm{E}[\mathsf{cost}]}{\mathbf{OPT}} \leq \frac{2}{\epsilon'}\left(1 + \frac{1}{\ln(1-\alpha)^{-1}}\right). \tag{7.9}$$

7.7 Set Cover

We now turn our attention to private approximations for the Set Cover: as for vertex cover, we do not explicitly output a set cover, but instead output a permutation over the sets in the set system so that each universe element is deemed covered by the first set in this permutation that contains it. Our algorithms for set cover give the slightly weaker (ϵ, δ)-privacy guarantees.

7.7.1 Unweighted Set Cover

We are given a set system $(U, \mathcal{S})$ and must cover a private subset $R \subset U$. Let the cardinality of the set system be $|\mathcal{S}| = m$, and let $|U| = n$. We first observe a computationally inefficient algorithm.

Theorem 53. *The exponential mechanism, when used to pick a permutation of sets, runs in time $O(m! poly(n))$ and gives an $O(\log(em/\mathbf{OPT})/\epsilon)$-approximation.*

Proof. A random permutation, with probability at least $\binom{m}{\mathbf{OPT}}^{-1}$ has all the sets in $\mathbf{OPT}$ before any set in $\mathbf{OPT}^c$. Thus the additive error is $O(\log \binom{m}{\mathbf{OPT}}/\epsilon)$. □

The rest of the section gives a computationally efficient algorithm with slightly worse guarantees: this is a modified version of the greedy algorithm, using the exponential mechanism to bias towards picking large sets.

Algorithm 5 Unweighted Set Cover

Input: Set system $(U, \mathcal{S})$, private $R \subset U$ of elements to cover, ϵ,δ.
let $i \leftarrow 1$, $R_i = R$. $\epsilon' \leftarrow \epsilon/2\ln(\frac{e}{\delta})$.
while not all sets in $\mathcal{S}$ have been output **do**
  pick a set S from $\mathcal{S}$ with probability proportional to $\exp(\epsilon'|S \cap R_i|)$.
  output set S.
  $R_{i+1} \leftarrow R_i \setminus S$, $i \leftarrow i + 1$.
end while

Utility Analysis

At the beginning of iteration i, say there are $m_i = m - i + 1$ remaining sets and $n_i = |R_i|$ remaining elements, and define $L_i = \max_{S \in \mathcal{S}} |S \cap R_i|$, the largest number of uncovered elements covered by any set in $\mathcal{S}$. By a standard argument, any algorithm that always picks sets of size $L_i/2$ is an $O(\ln n)$ approximation algorithm.

Theorem 54. *The above algorithm achieves an expected approximation ratio of $O(\ln n + \frac{\ln m}{\epsilon'}) = O(\ln n + \frac{\ln m \ln(e/\delta)}{\epsilon})$.*

Proof. As there is at least one set containing L_i elements, our use of the exponential mechanism to select sets combined with Equation 7.2 ensures that the probability we select a set covering fewer than $L_i - 3\ln m/\epsilon$ elements is at most $1/m^2$. While $L_i > 6\ln m/\epsilon$, with

probability at least $(1 - 1/m)$ we always select sets that cover at least $L_i/2$ elements, and can therefore use no more than $O(\mathbf{OPT} \ln n)$ sets. Once L_i drops below this bound, we observe that the number of remaining elements $|R_i|$ is at most $\mathbf{OPT} \cdot L_i$. Any permutation therefore costs at most an additional $O(\mathbf{OPT} \ln m/\epsilon')$. □

Privacy

Theorem 55. *The unweighted set cover algorithm preserves (ϵ, δ) differential privacy for any $\epsilon \in (0, 1)$, and $\delta < 1/e$.*

Proof. Let A and B be two set cover instances that differ in some element I. Say that S^I is the collection of sets containing I. Fix an output permutation π, and write $s_{i,j}(A)$ to denote the size of set S_j after the first $i - 1$ sets in π have been added to the cover.

$$\begin{aligned}\frac{\Pr[M(A) = \pi]}{\Pr[M(B) = \pi]} &= \prod_{i=1}^{n} \left(\frac{\exp(\epsilon' \cdot s_{i,\pi_i}(A))/(\sum_j \exp(\epsilon' \cdot s_{i,j}(A)))}{\exp(\epsilon' \cdot s_{i,\pi_i}(B))/(\sum_j \exp(\epsilon' \cdot s_{i,j}(B)))} \right) \\ &= \frac{\exp(\epsilon' \cdot s_{t,\pi_t}(A))}{\exp(\epsilon' \cdot s_{t,\pi_t}(B))} \cdot \prod_{i=1}^{t} \left(\frac{\sum_j \exp(\epsilon' \cdot s_{i,j}(B))}{\sum_j \exp(\epsilon' \cdot s_{i,j}(A))} \right)\end{aligned}$$

where t is such that S_{π_t} is the first set containing I to fall in the permutation π. After t, the remaining elements in A and B are identical, and all subsequent terms cancel. Moreover, except for the t^{th} term, the numerators of both the top and bottom expression cancel, since all the relevant set sizes are equal. If A contains I and B does not the first term is $\exp(\epsilon')$ and the each term in the product is at most 1.

Now suppose that B contains I and A does not . In this case, the first term is $\exp(-\epsilon') < 1$. Moreover, in instance B, every set in S^I is larger by 1 than in A, and all others remain the same size. Therefore, we have:

$$\begin{aligned}\frac{\Pr[M(A) = \pi]}{\Pr[M(B) = \pi]} &\leq \prod_{i=1}^{t} \left(\frac{(\exp(\epsilon') - 1) \cdot \sum_{j \in S^I} \exp(\epsilon' \cdot s_{i,j}(A)) + \sum_j \exp(\epsilon' \cdot s_{i,j}(A))}{\sum_j \exp(\epsilon' \cdot s_{i,j}(A))} \right) \\ &= \prod_{i=1}^{t} (1 + (\exp(\epsilon') - 1) \cdot p_i(A))\end{aligned}$$

where $p_i(A)$ is the probability that a set containing I is chosen at step i of the algorithm running on instance A, conditioned on picking the sets $S_{\pi_1}, \ldots, S_{\pi_{i-1}}$ in the previous steps.

By the definition of t, no set containing I has been chosen in the first $t - 1$ rounds; since at each round $i \leq t$, the probability of not choosing I is at most $1 - p_i(A)$ (it is even

smaller on instance B), the probability that I is not chosen in the first $t-1$ rounds is at most $\prod_{i=1}^{t-1}(1-p_i(A)) \leq \exp(-\sum_{i=1}^{t-1} p_i(A))$. Setting this quantity to be less than δ, we have that with probability at least $1-\delta$,

$$\sum_{i=1}^{t-1} p_i(A) \leq \ln \delta^{-1}.$$

Conditioning on this event, we can continue the analysis from above:

$$\begin{aligned}\frac{\Pr[M(A)=\pi]}{\Pr[M(B)=\pi]} &\leq \prod_{i=1}^{t} \exp((\exp(\epsilon')-1)p_i(A)) \leq \exp(2\epsilon' \sum_{i=1}^{t} p_i(A)) \\ &\leq \exp(2\epsilon'(\ln(\frac{1}{\delta}) + p_t(A))) \leq \exp(2\epsilon'(\ln(\frac{1}{\delta})+1)).\end{aligned}$$

Thus except with probability $(1-\delta)$, the affect of I on the distribution is at most $exp(\epsilon)$. □

Corollary 56. *For $\epsilon < 1$ and $\delta = 1/\mathrm{poly}(n)$, there is an $O(\frac{\ln n \ln m}{\epsilon})$-approximation algorithm for the unweighted set cover problem preserving (ϵ, δ)-differential privacy.*

7.7.2 Weighted Set Cover

We are given a set system $(U, \mathcal{S})$ and a cost function $C : \mathcal{S} \to \mathbb{R}$. We must cover a private subset $R \subset U$. W.l.o.g., let $\min_{S\in\mathcal{S}} C(S) = 1$, and denote $\max_{S\in\mathcal{S}} C(S) = W$. Let the cardinality of the set system be $|\mathcal{S}| = m$, and let $|U| = n$.

Algorithm 6 Weighted Set Cover

let $i \leftarrow 1$, $R_i = R$, $\mathcal{S}_i \leftarrow \mathcal{S}$, $r_i \leftarrow n$, $\epsilon' = \frac{\epsilon}{2\ln(e/\delta)}$, $T = \Theta\big(\frac{\log m + \log\log(nW)}{\epsilon'}\big)$
while $r_i \geq 1/W$ **do**
  pick a set S from $\mathcal{S}_i$ with probability proportional to $\exp\big(\epsilon'\big(\,|S\cap R_i| - r_i \cdot C(S)\,\big)\big)$
    or `halve` with probability proportional to $\exp(-\epsilon' T)$
  if `halve` **then**
    let $r_{i+1} \leftarrow r_i/2$, $R_{i+1} \leftarrow R_i$, $\mathcal{S}_{i+1} \leftarrow \mathcal{S}_i$, $i \leftarrow i+1$
  else
    output set S
    let $R_{i+1} \leftarrow R_i \setminus S$, $\mathcal{S}_{i+1} \leftarrow \mathcal{S}_i - \{S\}$, $r_{i+1} \leftarrow r_i$, $i \leftarrow i+1$
  end if
end while
output all remaining sets in $\mathcal{S}_i$ in random order

Let us first analyze the utility of the algorithm. If $R = \emptyset$, the algorithm has cost zero and there is nothing to prove. So we can assume that $\mathbf{OPT} \geq 1$. We first show that (**whp**) $r_i \gtrapprox R_i/\mathbf{OPT}$.

Lemma 20. *Except with probability* $1/\operatorname{poly}(m)$*, we have* $r_i \geq \frac{|R_i|}{2\mathbf{OPT}}$ *for all iterations* i*.*

Proof. Clearly $r_1 = n \geq |R_1|/2\mathbf{OPT}$. Since for r_i to fall below $|R_i|/2$, it must be in $(\frac{|R_i|}{2\mathbf{OPT}}, \frac{|R_i|}{\mathbf{OPT}}]$ and be halved in Step 6 of some iteration i. We'll show that this is unlikely: if at some iteration i, $\frac{|R_i|}{2\mathbf{OPT}} \leq r_i \leq \frac{|R_i|}{\mathbf{OPT}}$, then with high probability, the algorithm will not output `halve` and thus not halve r_i. Since all remaining elements R_i can be covered at cost at most $\mathbf{OPT}$, there must exist a set S such that $\frac{|S \cap R_i|}{C(S)} \geq \frac{|R_i|}{\mathbf{OPT}}$, and hence $|S \cap R_i| \geq C(S) \cdot \frac{|R_i|}{\mathbf{OPT}} \geq C(S) \cdot r_i$.

Hence $u_i(S) := |S \cap R_i| - r_i \cdot C(S) \geq 0$ in this case, and the algorithm will output S with probability at least proportional to 1, whereas it outputs `halve` with probability proportional to $\exp(-\epsilon' T)$. Thus, $\Pr[$ algorithm returns `halve` $] < \exp(-\epsilon' T) = 1/\operatorname{poly}(m \log nW)$. Since there are m sets in total, and r ranges from n to $1/W$, there are at most $m + O(\log nW)$ iterations, and the proof follows by a union bound. □

Let us define a *score* function $u_i(S) := |S \cap R_i| - r_i \cdot C(S)$, and $u_i(\texttt{halve}) := -T$: note that in Step 4 of our algorithm, we output either `halve` or a set S, with probabilities proportional to $\exp(\epsilon' u_i(\cdot))$. The following lemma states that with high probability, none of the sets output by our algorithm have very low scores (since we are much more likely to output `halve` than a low-scoring set).

Lemma 21. *Except with probability at most* $1/\operatorname{poly}(m)$*, Step 4 only returns sets* S *with* $u_i(S) \geq -2T$*.*

Proof. There are at most $|\mathcal{S}_i| \leq m$ sets S with score $u_i(S) \leq -2T$, and so one is output with probability at most proportional to $m \exp(-2T\epsilon)$. We will denote this bad event by $\mathcal{B}$. On the other hand, `halve` is output with probability proportional to $\exp(-T\epsilon)$. Hence, $\Pr[\texttt{halve}]/\Pr[\mathcal{B}] \geq \exp(T\epsilon)/m$, and so $\Pr[\mathcal{B}] \leq m/\exp(T\epsilon) \leq 1/\operatorname{poly}(m \log nW)$. Again there are at most $m + O(\log nW)$ iterations, and the lemma follows by a trivial union bound. □

We now analyze the cost incurred by the algorithm in each stage. Let us divide the algorithm's execution into *stages*: stage j consists of all iterations i where $|R_i| \in (\frac{n}{2^j}, \frac{n}{2^{j-1}}]$. Call a set S *interesting* if it is incident on an uncovered element when it is picked. Let $\mathcal{I}_j$ be the set of interesting sets selected in stage j, and $C(\mathcal{I}_j)$ be the total cost incurred on these sets.

Lemma 22. *Consider stages $1, \ldots, j$ of the algorithm. Let i^* be the index of the last iteration in stage j, and let $S_1, S_2, \ldots, S_{i^*}$ be the sets selected in the first i^* iterations of the algorithm. Then except with probability $1/\operatorname{poly}(m)$, we can bound the cost of the interesting sets in stage j by:*

$$\sum_{j' \le j} C(\mathcal{I}_{j'}) \le 4j\mathbf{OPT} \cdot (1 + 2T).$$

Proof. By Lemma 21 all the output sets have $u_i(S_i) \ge -2T$ **whp** . Rewriting, each S_i selected in a round $j' \le j$ satisfies

$$C(S_i) \le \frac{|S_i \cap R_i| + 2T}{r_i} \le \frac{2^{j'+1}\,\mathbf{OPT}}{n}(|S_i \cap R_i| + 2T),$$

where the second inequality is **whp** , and uses Lemma 20. Now summing over all rounds $j' \le j$, we get

$$\sum_{j' \le j} C(\mathcal{I}_{j'}) \le \sum_{j' \le j} \frac{2^{j'+1}\,\mathbf{OPT}}{n}\Big(\sum_{i \text{ s.t. } S_i \in \mathcal{I}_{j'}} \big(|S_i \cap R_i| + 2T\big)\Big).$$

Consider the inner sum for any particular value of j': let the first iteration in stage j' be iteration i_0—naturally $R_i \subseteq R_{i_0}$ for any iteration i in this stage. Now, since $S_i \cap R_i \subseteq R_{i_0}$ and $S_i \cap R_i$ is disjoint from $S_{i'} \cap R_{i'}$, the sum over $|S_i \cap R_i|$ is at most $|R_{i_0}|$, which is at most $\frac{n}{2^{j'-1}}$ by definition of stage j'. Moreover, since we are only concerned with bounding the cost of interesting sets, each $|S_i \cap R_i| \ge 1$, and so $|S_i \cap R_i| + 2T \le |S_i \cap R_i|(1 + 2T)$. Putting this together, (7.7.2) implies

$$\sum_{j' \le j} C(\mathcal{I}_{j'}) \le \sum_{j' \le j} \frac{2^{j'+1}\,\mathbf{OPT}}{n} \times \frac{n}{2^{j'-1}}(1 + 2T) = 4j\,\mathbf{OPT}\,(1 + 2T),$$

which proves the lemma. □

Theorem 57 (Utility). *The weighted set cover algorithm incurs a cost of $O(T \log n\,\mathbf{OPT})$ except with probability $1/\operatorname{poly}(m)$.*

Proof. Since the number of uncovered elements halves in each stage by definition, there are at most $1 + \log n$ stages, which by Lemma 22 incur a total cost of at most $O(\log n\ \mathbf{OPT} \cdot (1 + 2T))$. The sets that remain and are output at the very end of the algorithm incur cost at most W for each remaining uncovered element; since $r_i < 1/W$ at the end, Lemma 20 implies that $|R_i| < 2\mathbf{OPT}/W$ (**whp**), giving an additional cost of at most $2\,\mathbf{OPT}$. □

We can adapt the above argument to bound the expected cost by $O(T \log n \ \mathbf{OPT})$. (Proof in the full version.)

Theorem 58 (Privacy). *For any $\delta > 0$, the weighted set cover algorithm preserves (ϵ, δ) differential privacy.*

Proof. We imagine that the algorithm outputs a set named "HALVE" when Step 4 of the algorithm returns `halve`, and show that even this output is privacy preserving. Let A and B be two set cover instances that differ in some element I. Say that S^I is the collection of sets containing I. Fix an output π, and write $u_{i,j}(A)$ to denote the score of π_j (recall this may be `halve`) after the first $i-1$ sets in π have been selected.

$$\begin{aligned}\frac{\Pr[M(A)=\pi]}{\Pr[M(B)=\pi]} &= \prod_{i=1}^{n}\left(\frac{\exp(\epsilon' \cdot u_{i,\pi_i}(A))/(\sum_j \exp(\epsilon' \cdot u_{i,j}(A)))}{\exp(\epsilon' \cdot u_{i,\pi_i}(B))/(\sum_j \exp(\epsilon' \cdot u_{i,j}(B)))}\right) \\ &= \frac{\exp(\epsilon' \cdot u_{t,\pi_t}(A))}{\exp(\epsilon' \cdot u_{t,\pi_t}(B))} \cdot \prod_{i=1}^{t}\left(\frac{\sum_j \exp(\epsilon' \cdot u_{i,j}(B))}{\sum_j \exp(\epsilon' \cdot u_{i,j}(A))}\right)\end{aligned}$$

where t is such that S_{π_t} is the first set containing I to fall in the permutation π. After t, the remaining elements in A and B are identical, and all subsequent terms cancel. Moreover, except for the t^{th} term, the numerators of both the top and bottom expression cancel, since all the relevant set sizes are equal. If A contains I and B does not the first term is $\exp(\epsilon')$ and the each term in the product is at most 1.

Now suppose that B contains I and A does not . In this case, the first term is $\exp(-\epsilon') < 1$. Moreover, in instance B, every set in S^I is larger by 1 than in A, and all others remain the same size. Therefore, we have:

$$\begin{aligned}\frac{\Pr[M(A)=\pi]}{\Pr[M(B)=\pi]} &\leq \prod_{i=1}^{t}\left(\frac{(\exp(\epsilon')-1)\cdot\sum_{j\in S^I}\exp(\epsilon' \cdot u_{i,j}(A)) + \sum_j \exp(\epsilon' \cdot u_{i,j}(A))}{\sum_j \exp(\epsilon' \cdot u_{i,j}(A))}\right) \\ &= \prod_{i=1}^{t}\Big(1 + (e^{\epsilon'}-1)\cdot p_i(A)\Big)\end{aligned}$$

where $p_i(A)$ is the probability that a set containing I is chosen at step i of the algorithm running on instance A, conditioned on picking the sets $S_{\pi_1}, \ldots, S_{\pi_{i-1}}$ in the previous steps.

By the definition of t, no set containing I has been chosen in the first $t-1$ rounds; since at each round $i \leq t$, the probability of not choosing I is at most $1 - p_i(A)$ (it is even smaller on instance B), the probability that I is not chosen in the first $t-1$ rounds is at most $\prod_{i=1}^{t-1}(1-p_i(A)) \leq \exp(-\sum_{i=1}^{t-1} p_i(A))$.

There are two cases: If $\exp(-\sum_{i=1}^{t-1} p_i(A)) \leq \delta$, certainly $\Pr[M(A) = \pi] \leq \Pr[M(B) = \pi] + \delta$, which completes the proof. Otherwise,

$$\sum_{i=1}^{t-1} p_i(A) \leq \ln \delta^{-1}.$$

Continuing the analysis from above,

$$\begin{aligned}\frac{\Pr[M(A) = \pi]}{\Pr[M(B) = \pi]} &\leq \prod_{i=1}^{t} \exp((\exp(\epsilon') - 1)p_i(A)) \leq \exp\left(2\epsilon' \sum_{i=1}^{t} p_i(A)\right) \\ &\leq \exp\left(2\epsilon'\left(\ln\left(\frac{1}{\delta}\right) + p_t(A)\right)\right) \leq \exp\left(2\epsilon'\left(\ln\left(\frac{1}{\delta}\right) + 1\right)\right).\end{aligned}$$

Thus, in this second case, as desired, the effect of I on the output distribution is at most a multiplicative $\exp(\epsilon)$. □

7.7.3 Removing the Dependence on W

We can remove the dependence of the algorithm on W with a simple idea. For an instance $\mathcal{I} = (U, \mathcal{S})$, let $\mathcal{S}^j = \{S \in \mathcal{S} \mid C(S) \in (n^j, n^{j+1}]\}$. Let U^j be the set of elements such that the cheapest set containing them is in $\mathcal{S}^j$. Suppose that for each j and each $S \in \mathcal{S}^j$, we remove all elements that can be covered by a set of cost at most n^{j-1}, and hence define S' to be $S \cap (U^j \cup U^{j-1})$. This would change the cost of the optimal solution only by a factor of 2, since if we were earlier using S in the optimal solution, we can pick S' and at most n sets of cost at most n^{j-1} to cover the elements covered by $S \setminus S'$. Call this instance $\mathcal{I}' = (U, \mathcal{S}')$.

Now we partition this instance into two instances $\mathcal{I}_1$ and $\mathcal{I}_2$, where $\mathcal{I}_1 = (\cup_{j \text{ even}} U^j, \mathcal{S}')$, and where $\mathcal{I}_2 = (\cup_{j \text{ odd}} U^j, \mathcal{S}')$. Since we have just partitioned the universe, the optimal solution on both these instances costs at most $2\,\mathbf{OPT}(\mathcal{I})$. But both these instances $\mathcal{I}_1, \mathcal{I}_2$ are themselves collections of *disjoint* instances, with each of these instances having $w_{\max}/w_{\min} \leq n^2$; this immediately allows us to remove the dependence on W. Note that this transformation is based only on the set system $(U, \mathcal{S})$, and not on the private subset R.

Theorem 59. *For any $\epsilon \in (0, 1)$, $\delta = 1/\operatorname{poly}(n)$, there is an $O(log^2 n(\log m + \log\log n)/\epsilon)$-approximation for the weighted set cover problem that preserves (ϵ, δ)-differential privacy.*

7.7.4 Lower bounds

Theorem 60. *Any ϵ-differentially private algorithm that maps elements to sets must have approximation factor $\Omega(\log m/\epsilon)$, for a set cover instance with m sets and $((\log m)/\epsilon)^{O(1)}$ elements, for any $\epsilon \in (2\log m/m^{\frac{1}{20}}, 1)$.*

Proof. We consider a set system with $|U| = N$ and $\mathcal{S}$ a uniformly random selection of m size-k subsets of U. We will consider problem instances S_i consisting of one of these m subsets, so $\mathbf{OPT}(S_i) = 1$.

Let M be an ϵ-differentially private algorithm that on input $T \subseteq U$, outputs an assignment f mapping each element in U to some set in $\mathcal{S}$ that covers it. The number of possible assignments is at most m^N. The cost on input T under an assignment f is the cardinality of the set $f(T) = \cup_{e\in T} f(e)$.

We say assignment f is good for a subset $T \subseteq U$ if its cost $|f(T)|$ is at most $l = \frac{k}{2}$. We first show that any fixed assignment $f : U \to [m]$, such that $|f^{-1}(j)| \leq k$ for all j, is unlikely to be good for a randomly picked size-k subset T of U.

The number of ways to choose l sets from among those with non-empty $f^{-1}(\cdot)$ is at most $\binom{N}{l}$. Thus the probability that f is good for a random size-k subset is at most $\binom{N}{l}\left(\frac{lk}{N}\right)^k$. Setting $k = N^{1/10}$, and $l = \frac{k}{2}$, this is at most

$$\left(\frac{Ne}{l}\right)^l \left(\frac{lk}{N}\right)^k = \left(\frac{ek^3}{2N}\right)^{k/2} \leq 2^{-k\log N/4}.$$

Let $m = 2^{2\epsilon k}$. The probability that f is good for at least t of our m randomly picked sets is bounded by

$$\binom{m}{t}\left(2^{-k\log N/4}\right)^t \leq 2^{2\epsilon kt} 2^{-tk\log N/4} \leq 2^{-tk\log k/8}.$$

Thus, with probability at most $2^{-Nk\log k/8}$, a fixed assignment is good for more than N of m randomly chosen size-k sets. Taking a union bound over $m^N = 2^{2\epsilon kN}$ possible assignments, the probability that *any* feasible assignment f is good for more than N sets is at most $2^{-Nk\log k/16}$. Thus there exists a selection of size-k sets $S_1, \ldots, S_m$ such that no feasible assignment f is good for more than N of the S_i's.

Let $p_{M(\emptyset)}(S_i)$ be the probability that an assignment drawn from the distribution defined by running M on the the empty set as input is good for S_i. Since any fixed assignment is good for at most N of the m sets, the average value of $p_{M(\emptyset)}$ is at most N/m. Thus

there exists a set, say S_1 such that $p_{M(\emptyset)}(S_1) \leq N/m$. Since $|S_i| = k$ and M is ϵ-differentially private, $p_{M(S_1)}(S_1) \leq exp(\epsilon k) p_{M(\emptyset)}(S_1) < \frac{1}{2}$. Thus with probability at least half, the assignment M picks on S_1 is not good for S_1. Since $\mathbf{OPT}(S_1) = 1$, the expected approximation ratio of M is at least $l/2 = \frac{\log m}{4\epsilon}$.

Additionally, one can take s distinct instances of the above problem, leading to a new instance on $s \cdot N$ elements and $s \cdot m$ sets. $\mathbf{OPT}$ is now s, while it is easy to check that any private algorithm must cost $\Omega(s \cdot l)$ in expectation. Thus the lower bound in fact rules out additive approximations. □

7.7.5 An Inefficient Algorithm for Weighted Set Cover

For completeness, we now show that the lower bound shown above is tight even in the weighted case, in the absence of computational constraints. Recall that we are given a collection $\mathcal{S}$ of subsets of a universe U, and a private subset $R \subseteq U$ of elements to be covered. Additionally, we have weights on sets; we round up weights to powers of 2, so that sets in $\mathcal{S}_j$ have weight exactly 2^{-j}. Without loss of generality, the largest weight is 1 and the smallest weight is $w = 2^{-L}$.

As before, we will output a permutation π on $\mathcal{S}$, with the understanding that the cost $cost(R, \pi)$ of a permutation π on input R is defined to be the total cost of the set cover resulting from picking the first set in the permutation containing e, for each $e \in R$.

Our algorithm constructs this permutation in a gradual manner. It maintains a permutation π_j on $\cup_{i \leq j} \mathcal{S}_i$ and a threshold T_j. In step j, given π_{j-1} and T_{j-1}, the algorithm constructs a partial permutation π_j on $\cup_{i \leq j} \mathcal{S}_j$, and a threshold T_j. In each step, we use the exponential mechanism to select an extension with an appropriate base distribution μ_j and score function q. At the end of step L, we get our permutation $\pi = \pi_L$ on $\mathcal{S}$.

Our permutations π_j will all have a specific structure. The weight of the ith set in the permutation, as a function of i will be a unimodal function that is non-increasing until T_j, and then non-decreasing. In other words, π_j contains sets from $\mathcal{S}_j$ as a continuous block. The sets that appear before T_j are said to be in the *bucket*. We call a partial permutation respecting this structure *good*. We say a good permutation π *extends* a good partial permutation π_j if π_j and π agree on their ordering on $\cup_{i \leq j} \mathcal{S}_i$.

We first define the score function that is used in these choices. A natural objective function would be $cost(R, \pi_j) = \min_{\pi \text{ extends } \pi_j} cost(R, \pi)$, i.e. the cost of the optimal solution conditioned on respecting the partial permutation π_j. We use a slight modification of this score function: we force the cover to contain all sets from the bucket and denote as $\widetilde{cost}(R, \pi)$ the resulting cover defined by R on π. We then define $\widetilde{cost}(R, \pi_j)$ naturally as

$\min_\pi$ extends $_{\pi_j}\ \widetilde{cost}(R,\pi)$. We first record the following easy facts:

Observation 61. *For any R, $\min_\pi \widetilde{cost}(R,\pi) = \min_\pi cost(R,\pi) = \mathbf{OPT}$. Moreover, for any π, $\widetilde{cost}(R,\pi) \geq cost(R,\pi)$.*

To get (π_j, T_j) given (π_{j-1}, T_{j-1}), we insert a permutation σ_j of $\mathcal{S}_j$ after the first T_{j-1} elements of π_{j-1}, and choose T_j, where both σ_j and T_j are chosen using the exponential mechanism. The base measure on σ_j is uniform and the base measure on $T_j - T_{j-1}$ is the geometric distribution with parameter $1/m^2$.

Let $\widetilde{cost}(A,(\sigma_j,T_j))$ be defined as $\widetilde{cost}(A,\pi_j) - \widetilde{cost}(A,\pi_{j-1})$, where π_j is constructed from π_{j-1} and (σ_j,T_j) as above. The score function we use to pick (σ_j,T_j) is $score_j(R,(\sigma_j,T_j)) = 2^j\widetilde{cost}(R,(\sigma_j,T_j))$. Thus $Pr[(\sigma_j,T_j)] \propto (1/m^2(T_j - T_{j-1}))\exp(\epsilon score((\sigma_j,T_j)))$.

Let the optimal solution to the instance contain n_j sets from $\mathcal{S}_j$. Thus $\mathbf{OPT} = \sum_j 2^{-j}n_j$. We first show that $\widetilde{cost}(R,\pi_L)$ is $O(\mathbf{OPT}\log m/\epsilon)$. By Observation 61, the approximation guarantee would follow.

The probability that the n_j sets in $\mathbf{OPT}$ fall in the bucket when picking from the base measure is at least $1/m^{3n_j}$. When that happens, $\widetilde{cost}(R,\pi_j) = \widetilde{cost}(R,\pi_{j-1})$. Thus the exponential mechanism ensures that except with probability $1/poly(m)$:

$$\widetilde{cost}(R,\pi_j) \leq \widetilde{cost}(R,\pi_{j-1}) + 4\cdot 2^{-j}\log(m^{3n_j})/\epsilon = \widetilde{cost}(R,\pi_{j-1}) + 12\cdot 2^{-j}n_j\log m/\epsilon$$

Thus with high probability,

$$\begin{aligned}\widetilde{cost}(R,\pi_L) &\leq \widetilde{cost}(R,\pi_0) + 12\sum_j 2^{-j}n_j\log m/\epsilon \\ &= \mathbf{OPT} + 12\mathbf{OPT}\log m/\epsilon\end{aligned}$$

Finally, we analyze the privacy. Let $e \in U$ be an element such that the cheapest set covering U has cost 2^{-j_e}. Let A and B be two instances that differ in element e. It is easy to see that $|\widetilde{cost}(A,(\sigma_j,T_j)) - \widetilde{cost}(B,(\sigma_j,T_j))|$ is bounded by 2^{-j} for all j. We show something stronger:

Lemma 23. *For any good partial permutation π_j and any A, B such that $A = B \cup \{e\}$,*

$$|score(A,(\sigma_j,T_j)) - score_j(B,(\sigma_j,T_j))| \leq \begin{cases} 0 & \text{if } j > j_e \\ 2^{j_e-j+1} & \text{if } j \leq j_e \end{cases}$$

Proof. Let π_B be the permutation realizing $\widetilde{cost}(B)$. For $j \leq j_e$, if e is covered by a set in the bucket in π_B, then the cost of π_B is no larger in instance A and hence $\widetilde{cost}(A, \pi_j) = \widetilde{cost}(B, \pi_j)$[1]. In the case that the bucket in π_B does not cover e, then $\widetilde{cost}(A, \pi_j) \leq \widetilde{cost}(A, \pi_B) = \widetilde{cost}(B, \pi_B) + 2^{-j_e} = \widetilde{cost}(B, \pi_j) + 2^{-j_e}$. Since this also holds for π_{j-1}, this implies the claim from $j \leq j_e$.

For $j > j_e$, observe that the first set in π_B that covers e is fully determined by the partial permutation π_j, since the sets in $\cup_{i>j_e} \mathcal{S}_i$ do not contain e. Thus $\widetilde{cost}(A, (\sigma_j, T_j)) = \widetilde{cost}(B, (\sigma_j, T_j))$ and the claim follows. □

Then for any $j \leq j_e$, lemma 23 implies that for any (σ_j, T_j), $\exp(\epsilon(score(A, (\sigma_j, T_j)) - score(B, (\sigma_j, T_j)))) \in [\exp(-\epsilon 2^{j_e - j+1}), \exp(\epsilon 2^{j_e-j+1})]$. Thus

$$\frac{\Pr[\sigma_j, T_j|A]}{\Pr[\sigma_j, T_j|B]} \in [\exp(-2^{j-j_e+2}\epsilon), \exp(2^{j-j_e+2}\epsilon)]$$

Moreover, for any $j > j_e$, this ratio is 1. Thus

$$\begin{aligned} \frac{\Pr[\sigma_j, T_j|A]}{\Pr[\sigma_j, T_j|B]} &\in [\Pi_{j \leq j_e} \exp(-2^{j-j_e+2}\epsilon), \Pi_{j \leq j_e} \exp(2^{j-j_e+2}\epsilon)] \\ &\subseteq [\exp(-8\epsilon), \exp(8\epsilon)], \end{aligned}$$

which implies 8ϵ-differential privacy.

7.8 Combinatorial Public Projects (Submodular Maximization)

Recently Papadimitriou et al.[PSS08] introduced the Combinatorial Public Projects Problem (CPP Problem) and showed that there is a succinctly representable version of the problem for which, although there exists a constant factor approximation algorithm, no efficient *truthful* algorithm can guarantee an approximation ratio better than $m^{\frac{1}{2}-\epsilon}$, unless $NP \subseteq BPP$. Here we adapt our set cover algorithm to give a privacy preserving approximation to the CPP problem within logarithmic (additive) factors.

In the CPP problem, we have n agents and m resources publicly known. Each agent submits a private non-decreasing and *submodular* valuation function f_i over subsets of resources, and our goal is to select a size-k subset S of the resources to maximize

[1]We remark that this is not true for the function $cost$, and is the reason we had to modify it to $\widetilde{cost}$.

$\sum_{i=1}^{n} f_i(S)$. We assume that we have oracle access to the functions f_i. Note that since each f_i is submodular, so is $\sum_{i=1}^{n} f_i(S)$, and our goal is to produce a algorithm for submodular maximization that preserves the privacy of the individual agent valuation functions. Without loss of generality, we will scale the valuation functions such that they take maximum value 1: $\max_{i,S} f_i(S) = 1$.

Once again, we have an easy computationally inefficient algorithm.

Theorem 62. *The exponential mechanism when used to choose k sets runs in time $O(\binom{m}{k} poly(n))$ and has expected quality at least $OPT - O(\log \binom{m}{k}/\epsilon)$.*

We next give a computationally efficient algorithm with slightly worse guarantees. We adapt our unweighted set cover algorithm, simply selecting k items greedily:

Algorithm 7 CPP Problem

1: **Input:** A set of M of m resources, private functions $f_1, \ldots, f_n$, a number of resources k, ϵ, δ.
2: **let** $M_1 \leftarrow M$, $F(x) := \sum_{i=1}^{m} f_i(x)$, $S_1 \leftarrow \emptyset$, $\epsilon' \leftarrow \frac{\epsilon}{8e \ln(2/\delta)}$.
3: **for** $i = 1$ to k **do**
4: **pick** a resource r from M_i with probability proportional to $\exp(\epsilon'(F(S_i + \{r\}) - F(S_i)))$.
5: **let** $M_{i+1} \leftarrow M_i - \{r\}$, $S_{i+1} \leftarrow S_i + \{r\}$.
6: **end for**
7: **Output** S_{k+1}.

7.8.1 Utility Analysis

Theorem 63. *Except with probability $O(1/poly(n))$, the algorithm for the CPP problem returns a solution with quality at least $(1 - 1/e)\mathbf{OPT} - O(k \log m/\epsilon')$.*

Proof. Since F is submodular and there exists a set S^* with $|S| = k$ and $F(S) = \mathbf{OPT}$, there always exists a resource r such that $F(S_i + \{r\}) - F(S_i) \geq (\mathbf{OPT} - F(S_i))/k$. If we always selected the optimizing resource, the distance to $\mathbf{OPT}$ would decrease by a factor of $1 - 1/k$ each round, and we would achieve an approximation factor of $1 - 1/e$. Instead, we use the exponential mechanism which, by (7.2), selects a resource within $4 \ln m/\epsilon'$ of the optimizing resource with probability at least $1 - 1/m^3$. With probability at least $1 - k/m^3$ each of the k selections decreases $\mathbf{OPT} - F(S_i)$ by a factor of $(1 - 1/k)$, while increasing it by at most an additive $4 \ln m/\epsilon'$, giving $(1 - 1/e)\mathbf{OPT} + O(k \ln m/\epsilon')$.

□

7.8.2 Privacy Analysis

Theorem 64. *For any $\delta \leq 1/2$, the CPP problem algorithm preserves $(8\epsilon'(e-1)\ln(2/\delta), \delta)$-differential privacy.*

Proof. Let A and B be two CPP instances that differ in a single agent I with utility function f_I. We show that the output set of resources, even revealing the order in which the resources were chosen, is privacy preserving. Fix some ordered set of k resources, $\pi_1, \ldots, \pi_k$ write $S_i = \bigcup_{j=1}^{i-1}\{\pi(j)\}$ to denote the first $i-1$ elements, and write $s_{i,j}(A) = F_A(S_i + \{j\}) - F_A(S_i)$ to denote the marginal utility of item j at time i in instance A. Define $s_{i,j}(B)$ similarly for instance B. We consider the relative probability of our mechanism outputting ordering π when given inputs A and B:

$$\frac{\Pr[M(A)=\pi]}{\Pr[M(B)=\pi]} = \prod_{i=1}^{k}\left(\frac{\exp(\epsilon' \cdot s_{i,\pi_i}(A))/(\sum_j \exp(\epsilon' \cdot s_{i,j}(A)))}{\exp(\epsilon' \cdot s_{i,\pi_i}(B))/(\sum_j \exp(\epsilon' \cdot s_{i,j}(B)))}\right),$$

where the sum over j is over all remaining unselected resources. We can separate this into two products

$$\prod_{i=1}^{k}\left(\frac{\exp(\epsilon' \cdot s_{i,\pi_i}(A))}{\exp(\epsilon' \cdot s_{i,\pi_i}(B))}\right) \cdot \prod_{i=1}^{k}\left(\frac{\sum_j \exp(\epsilon' \cdot s_{i,j}(B))}{\sum_j \exp(\epsilon' \cdot s_{i,j}(A))}\right).$$

If A contains agent I but B does not, the second product is at most 1, and the first is at most $\exp(\epsilon' \sum_{i=1}^{k}(F_I(S_i) - F_I(S_{i-1}))) \leq \exp(\epsilon')$. If B contains agent I, and A does not, the first product is at most 1, and in the remainder of the proof, we focus on this case. We will write $\beta_{i,j} = s_{i,j}(B) - s_{i,j}(A)$ to be the additional marginal utility of item j at time i in instance B over instance A, due to agent I. Thus

$$\begin{aligned}
\frac{\Pr[M(A)=\pi]}{\Pr[M(B)=\pi]} &\leq \prod_{i=1}^{k}\left(\frac{\sum_j \exp(\epsilon' \cdot s_{i,j}(B))}{\sum_j \exp(\epsilon' \cdot s_{i,j}(A))}\right) \\
&= \prod_{i=1}^{k}\left(\frac{\sum_j \exp(\epsilon'\beta_{i,j}) \cdot \exp(\epsilon' \cdot s_{i,j}(A))}{\sum_j \exp(\epsilon' \cdot s_{i,j}(A))}\right) \\
&= \prod_{i=1}^{k} \mathrm{E}_i[\exp(\epsilon'\beta_i)],
\end{aligned}$$

where β_i is the marginal utility actually achieved at time i by agent I, and the expectation is taken over the probability distribution over resources selected at time i in instance A. For all $x \leq 1$, $e^x \leq 1 + (e-1) \cdot x$. Therefore, for all $\epsilon' \leq 1$, we have:

$$\begin{aligned}\prod_{i=1}^{k} \mathrm{E}_i[\exp(\epsilon'\beta_i)] &\leq \prod_{i=1}^{k} E_i[1 + (e-1)\epsilon'\beta_i] \\ &\leq \exp((e-1)\epsilon' \sum_{i=1}^{k} E_i[\beta_i]).\end{aligned}$$

We must therefore show that with high probability, the sum of expected marginal utilities of agent I is small, which we will denote by $C = \sum_{i=1}^{k} E_i[\beta_i]$.

We note that the *realized* total utility of agent i is at most 1: $\sum_{i=1}^{k} \beta_{i,\pi_i} \leq 1$. If C is large, then the realized utility of agent i has a large deviation from its expected value; such realizations π must be rare. By a Chernoff bound, this occurs only with small probability:

$$\Pr[|\sum_{i=1}^{k} \beta_{i,\pi_i} - C| \geq C - 1] \leq 2\exp\left(\frac{-(C-1)^2}{3C-1}\right).$$

Setting this probability to be at most δ and solving for C, we find that except with probability δ, $C \leq 8\ln(2/\delta)$. Therefore, with probability $1-\delta$ we have:

$$\frac{\Pr[M(A) = \pi]}{\Pr[M(B) = \pi]} \leq \exp((e-1)\epsilon' \cdot 8\ln\left(\frac{2}{\delta}\right))$$

which proves the theorem.

□

Remark: By choosing $\epsilon' = \epsilon/k$, we immediately get ϵ-differential privacy and expected utility at least $\mathbf{OPT} - O(k^2 \ln m/\epsilon)$. This may give better guarantees for some values of k and δ.

We remark that the k-coverage problem is a special case of the CPP problem. Therefore:

Corollary 65. *The CPP algorithm (with* sets *as resources) is an (ϵ, δ)-differential privacy preserving algorithm for the k-coverage problem achieving approximation factor at least $(1-1/e)\mathbf{OPT} - O(k \log m \log(2/\delta)/\epsilon)$.*

7.8.3 Truthfulness

The CPP problem can be viewed as a mechanism design problem when each agent i has a choice of whether to submit his actual valuation function f_i, or to lie and submit a different valuation function f_i' if such a misrepresentation yields a better outcome for agent i. A mechanism is *truthful* if for every valuation function of agents $j \neq i$, and every valuation function f_i of agent i, there is never a function $f_i' \neq f_i$ such that agent i can benefit by misrepresenting his valuation function as f_i'. Intuitively, a mechanism is approximately truthful if no agent can make more than a slight gain by not truthfully reporting.

Definition 28. *A mechanism for the CPP problem is γ-truthful if for every agent i, for every set of player valuations f_j for $j \neq i$, and for every valuation function $f_i' \neq f_i$:*

$$E[f_i(M(f_1, \ldots, f_i, \ldots, f_n))] \geq E[f_i(M(f_1, \ldots, f_i', \ldots, f_n))] - \gamma$$

Note that 0-truthfulness corresponds to the usual notion of (exact) truthfulness.

(ϵ, δ)-differential privacy in our setting immediately implies $(2\epsilon + \delta)$-approximate truthfulness.We note that Papadimitriou et al. [PSS08] showed that the CPP problem is inapproximable to an $m^{\frac{1}{2}-\epsilon}$ multiplicative factor by any polynomial time 0-truthful mechanism. Our result shows that relaxing that to γ-truthfulness allows us to give a constant approximation to the utility whenever $\mathbf{OPT} \geq 2k \log m \log(1/\gamma)/\gamma$ for any γ.

7.8.4 Lower Bounds

Theorem 66. *No ϵ-differentially private algorithm for the maximum coverage problem can guarantee profit larger than $\mathbf{OPT} - (k \log(m/k)/20\epsilon)$.*

The proof is almost identical to that of the lower bound Theorem 47 for k-median, and hence is omitted.

Bibliography

[AB99] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999. 3.1, 3.1.2, 1, 3.3

[ABDCBH97] N. Alon, S. Ben David, N. Cesa Bianchi, and D. Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM (JACM)*, 44(4):615–631, 1997. 5.1, 5.4

[AGK+04] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004. 7.1.1, 7.4, 44

[BBV06] M.F. Balcan, A. Blum, and S. Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 65(1):79–94, 2006. 10

[BCNW06] A. Beimel, P. Carmi, K. Nissim, and E. Weinreb. Private approximation of search problems. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 119–128. ACM New York, NY, USA, 2006. 7.1.2, 7.5

[BDMN05] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 128–138. ACM New York, NY, USA, 2005. 2.4, 3.1.1

[BFJ+94] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, page 262. ACM, 1994. 6.2, 34

[BHN07] A. Beimel, R. Hallak, and K. Nissim. Private Approximation of Clustering and Vertex Cover. *Theory of Cryptography Conference*, 4392:383, 2007. 7.1.2

[BKN10] A. Beimel, S. Kasiviswanathan, and K. Nissim. Bounds on the sample complexity for private learning and private data release. *Theory of Cryptography*, pages 437–454, 2010. 5.3

[BL95] P.L. Bartlett and P.M. Long. More theorems about scale-sensitive dimensions and learning. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 392–401. ACM, 1995. 5.4, 28

[BLR08] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 609–618. ACM, 2008. 1.1.1, 3, 4.1.2, 5.1

[BLW94] P.L. Bartlett, P.M. Long, and R.C. Williamson. Fat-shattering and the learnability of real-valued functions. In *Proceedings of the seventh annual conference on Computational learning theory*, pages 299–310. ACM, 1994. 5.1, 19

[BMNW07] A. Beimel, T. Malkin, K. Nissim, and E. Weinreb. How Should We Solve Search Problems Privately? In *CRYPTO*, volume 4622, page 31. Springer, 2007. 7.1.2

[BZ06] M. Barbaro and T. Zeller. A face is exposed for aol searcher no. 4417749, August 9 2006. 2.1

[CDM+05] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Proceedings of the 2nd Theory of Cryptography Conference*, pages 363–385, 2005. 1

[Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, pages 233–235, 1979. 7.1.1

[Dal77] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15:429–444, 1977. 2.1

[Dev86] L. Devroye. *Non-uniform random variate generation*. 1986. 4.4.3

[DFK91] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM (JACM)*, 38(1):1–17, 1991. 4.1.2, 4.4, 4.4.1, 4.4.1

[DG99] S. Dasgupta and A. Gupta. An elementary proof of the Johnson-Lindenstrauss Lemma. *International Computer Science Institute, Technical Report*, pages 99–006, 1999. 10

[DKM+06] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our Data, Ourselves: Privacy via Distributed Noise Generation. *Proceedings of Advances in CryptologyEurocrypt 2006*, pages 486–503, 2006. 2.4

[DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Theory of Cryptography Conference TCC*, volume 3876 of *Lecture Notes in Computer Science*, page 265. Springer, 2006. 1, 1.1.1, 2.2, 2, 2.2, 4, 2.2, 1, 2.4, 3.1, 3.1.1, 3.2, 3.7.1, 3.8.2, 3.8.2, 4.1, 4.1.3, 5.1, 18, 2, 7.1.1, 7.5

[DMT07] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of LP decoding. In *Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing*, page 94. ACM, 2007. 2.4, 3.1, 3.1.1, 5.1, 5.1.1

[DN03] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 202–210, 2003. 2.4, 3.1, 3.1.1, 3.2, 4.1, 5.1, 5.1.1

[DN04] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Proceedings of CRYPTO*, Lecture Notes in Computer Science, pages 528–544. Springer, 2004. 1

[DNR+09] C. Dwork, M. Naor, O. Reingold, G.N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing*, pages 381–390. ACM New York, NY, USA, 2009. 1.1.1, 2.4, 4.1, 3, 4.1.2, 4.1.3, 5, 4.5, 5.1, 1, 6.3

[DS10] C. Dwork and A. Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2), 2010. 2.4

[Dwo06] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, volume 4052 of *LECTURE NOTES IN COMPUTER SCIENCE*, page 1. Springer, 2006. 1, 2.1, 2.4

[Dwo08] C. Dwork. Differential privacy: A survey of results. In *Proceedings of Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008*, volume 4978 of *Lecture Notes in Computer Science*, page 1. Springer, 2008. 2.4

[DY08] C. Dwork and S. Yekhanin. New efficient attacks on statistical disclosure control mechanisms. *Advances in Cryptology–CRYPTO 2008*, pages 469–480, 2008. 2.4, 5.1, 5.1.1

[FF56] L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956. 7.1.1

[FFKN09] D. Feldman, A. Fiat, H. Kaplan, and K. Nissim. Private coresets. In *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing*, pages 361–370. ACM New York, NY, USA, 2009. 2.4, 5.1

[FIM+06] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Secure multiparty computation of approximations. *ACM Trans. Algorithms*, 2(3):435–472, 2006. 7.1.2

[GLM+10] A. Gupta, K. Ligett, F. McSherry, A. Roth, and K. Talwar. Differentially Private Approximation Algorithms. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2010. 1.1.2

[GRS09] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing*, pages 351–360. ACM New York, NY, USA, 2009. 4.1, 4.1.3

[HKKN01] S. Halevi, R. Krauthgamer, E. Kushilevitz, and K. Nissim. Private approximation of NP-hard functions. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 550–559. ACM New York, NY, USA, 2001. 7.1.1, 7.1.2, 7.5

[Hoc82] D.S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555, 1982. 7.1.1

[HT10] M. Hardt and K. Talwar. On the Geometry of Differential Privacy. In *The 42nd ACM Symposium on the Theory of Computing, 2010. STOC'10*, 2010. 2.4, 4.1, 5.1, 5.1.1, 5.4

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998. 3.6

[IW06] P. Indyk and D. Woodruff. Polylogarithmic Private Approximations and Efficient Matching. *Theory of Cryptography*, 3876:245, 2006. 7.1.2

[Joh74] D.S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9:256–278, 1974. 7.1.1

[Kar93] David R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (Austin, TX, 1993)*, pages 21–30, New York, 1993. ACM. 7.3, 16

[Kea98] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998. 1.1.1, 6.2, 36

[KLN+07] Shiva Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? Manuscript, 2007. 3.1, 3.1.1, 3.3

[KLN+08] S.P. Kasiviswanathan, H.K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What Can We Learn Privately? In *IEEE 49th Annual IEEE Symposium on Foundations of Computer Science, 2008. FOCS'08*, pages 531–540, 2008. 1.1.1, 2.4, 5.1, 6.1, 6.2, 21, 22, 23, 33, 6.2, 6.2, 36

[KRSU10] S. Kasiviswanathan, M. Rudelson, A. Smith, and J. Ullman. The Price of Privately Releasing Contingency Tables and the Spectra of Random Matrices with Correlated Rows. In *The 42nd ACM Symposium on the Theory of Computing, 2010. STOC'10*, 2010. 2.4, 5.1.1

[KS94] M.J. Kearns and R.E. Schapire. Efficient distribution-free learning of probabilistic concepts*. *Journal of Computer and System Sciences*, 48(3):464–497, 1994. 19

[MT07] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual Symposium on Foundations of Computer Science*, 2007. 2.2, 5, 2, 2.4, 3.3, 5, 4.1, 20, 30, 5.4, 7.1, 7.2.1, 7.2.1

[NRS07] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Annual ACM Symposium on Theory of Computing: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. Association for Computing Machinery, Inc, One Astor Plaza, 1515 Broadway, New York, NY, 10036-5701, USA,, 2007. 2.4, 4.1.3, 5.1.1

[NS08] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, volume 8, pages 111–125. Citeseer, 2008. 2.1

[NWF78] GL Nemhauser, LA Wolsey, and ML Fisher. An analysis of approximations for maximizing submodular set functionsI. *Mathematical Programming*, 14(1):265–294, 1978. 7.1.1

[Pit85] L. Pitt. A simple probabilistic approximation algorithm for vertex cover. Technical report, Yale University, 1985. 7.1.1, 7.5.1, 7.5.3

[PSS08] C. Papadimitriou, M. Schapira, and Y. Singer. On the Hardness of Being Truthful. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008. 1.1.2, 7.1.1, 7.8, 7.8.3

[RR10] A. Roth and T. Roughgarden. Interactive Privacy via the Median Mechanism. In *The 42nd ACM Symposium on the Theory of Computing, 2010. STOC'10*, 2010. 1.1.1, 5.1

[S$^+$02] L. Sweeney et al. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5):557–570, 2002. 2.1

[SS02] A. J. Smola and B. Schölkopf. *Learning with Kernels*. MIT Press, 2002. 3.1

[UV10] J. Ullman and S. Vadhan. PCPs and the Hardness of Generating Synthetic Data . *Manuscript*, 2010. 4.5, 5.1

[Vap96] V. Vapnik. Structure of statistical learning theory. *Computational Learning and Probabilistic Reasoning*, page 3, 1996. 4.3.1

[Vap98] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons Inc., 1998. 3.1, 3.1.2, 1