Private Data Release via Learning Thresholds

Moritz Hardt[*] Guy N. Rothblum[†] Rocco A. Servedio[‡]

Abstract

This work considers *computationally efficient* privacy-preserving data release. We study the task of analyzing a database containing sensitive information about individual participants. Given a set of statistical queries on the data, we want to release approximate answers to the queries while also guaranteeing *differential privacy*—protecting each participant's sensitive data.

Our focus is on computationally efficient data release algorithms; we seek algorithms whose running time is polynomial, or at least sub-exponential, in the data dimensionality. Our primary contribution is a *computationally efficient* reduction from differentially private data release for a class of counting queries, to learning thresholded sums of predicates from a related class.

We instantiate this general reduction with algorithms for learning thresholds, obtaining new results for differentially private data release. As two examples, taking $\{0,1\}^d$ to be the data domain (of dimension d), we obtain differentially private algorithms for:

1. Releasing all k-way conjunction counting queries (or k-way contingency tables). For any given k, the resulting data release algorithm has bounded error as long as the database is of size at least $d^{O\left(\sqrt{k \log(k \log d)}\right)}$ (ignoring the dependence on other parameters). The running time is polynomial in the database size. The best sub-exponential time algorithms known prior to our work required a database of size $\tilde{O}(d^{k/2})$ [Dwork McSherry Nissim and Smith 2006].
2. Releasing any family of counting queries that is specified by a constant depth AC^0 predicate. This algorithm releases accurate answers to a $(1-\gamma)$-fraction of the queries in the family. For any $\gamma \geqslant \text{quasipoly}(1/d)$, the algorithm has bounded error as long as the database is of size at least quasipoly(d) (again ignoring the dependence on other parameters). The running time is quasipoly(d).

The first learning algorithm uses techniques for representing thresholded sums of predicates as low-degree polynomial threshold functions. The second learning algorithm is based on a result of Jackson Klivans and Servedio [JKS 2002], and utilizes Fourier analysis of the database viewed as a function mapping queries to answers.

1 Introduction

This work considers privacy-preserving statistical analysis of sensitive data. In this setting, we wish to extract statistics from a database D that contains information about n individual participants. Each individual's data is a record in the *data domain* $\mathcal{U}$. We focus here on the offline (or non-interactive) setting, in which the information to be extracted is specified by a set $\mathcal{Q}$ of statistical queries. Each query $q \in \mathcal{Q}$ is a function mapping the database to a query answer, where in this work we focus on real-valued queries with range $[0,1]$. Our goal is *data release*: Extracting approximate answers to all the queries in the query set $\mathcal{Q}$.

An important concern in this setting is protecting the privacy of individuals whose sensitive data (e.g. medical or financial records) are being analyzed. Differential privacy [DMNS06] provides a rigorous notion of privacy protection, guaranteeing that each individual only has a small effect on the data release algorithm's output. A growing body of work explores the possibility of extracting rich statistics in a differentially private manner. One line of research [BLR08, DNR$^+$09, DRV10, RR10, HR10] has shown that differential privacy often permits surprisingly accurate statistics. These works put forward general algorithms and techniques for differentially private data analysis, but the algorithms have running time that is (at least) exponential in the dimensionality of the data domain. Thus,

[*] Center for Computational Intractability, Department of Computer Science, Princeton University. Supported by NSF grants CCF-0426582 and CCF-0832797. Email: `mhardt@cs.princeton.edu`.

[†] Microsoft Research, Silicon Valley Campus. Most of this work was done while the author was at the Department of Computer Science at Princeton University and Supported by NSF Grant CCF-0832797 and by a Computing Innovation Fellowship. Email: `rothblum@alum.mit.edu`.

[‡] Columbia University Department of Computer Science and Center for Computational Intractability, Department of Computer Science, Princeton University. Supported by NSF grants CCF-0832797, CNS-07-16245 and CCF-0915929. Email: `rocco@cs.columbia.edu`

a central question in differentially private data analysis is to develop general techniques and algorithms that are *efficient*, i.e. with running time that is polynomial (or at least sub-exponential) in the data dimensionality. While some computational hardness results are known [DNR$^+$09, UV11, GHRU11], they apply only to restricted classes of data release algorithms.

This Work. Our primary contribution is a computationally efficient new tool for privacy-preserving data release: a general reduction to the task of *learning thresholds of sums of predicates.* The class of predicates (for learning) in our reduction is derived directly from the class of queries (for data release).

At a high level, we draw a connection between data release and learning as follows. In the data release setting, one can view the *database as a function*: it maps queries in $\mathcal{Q}$ to answers in $[0, 1]$. The data release goal is approximating this function on queries/examples in $\mathcal{Q}$. The challenge is doing so with only bounded access to the database/function; in particular, we only allow access that preserves differential privacy. For example, this often means that we only get a bounded number of oracle queries to the database function with noisy answers.

At this high level there is a striking similarity to learning theory, where a standard goal is to efficiently learn/approximate a function given limited access to it, e.g. a bounded number of labeled examples or oracle queries. Thus a natural approach to data release is *learning the database function* using a computational learning algorithm. Before proceeding, we note that previous works drew similar connections between private data release and learning theory. The works [DNR$^+$09, DRV10, GHRU11] are perhaps closest in spirit to ours. We elaborate on related works below.

While the approach outlined above is intuitively appealing at a high level, it faces immediate obstacles because of apparent incompatibilities between the requirements of learning algorithms and the type of "limited" access to data that are imposed by private data release. For example, in the data release setting a standard technique for ensuring differential privacy is adding noise, but many efficient learning algorithms fail badly when run on noisy data. As another example, for private data release, the number of (noisy) database accesses is often very restricted: e.g sub-linear, or at most quadratic in the database size. In the learning setting, on the other hand, it is almost always the case that the number of examples or oracle queries required to learn a function is *lower bounded* by its description length (and is often a large polynomial in the description length).

Our work explores the connection between learning and private data release. We

(i) give an efficient reduction that shows that, in fact, a general class of data release tasks *can* be reduced to related and natural computational learning tasks; and

(ii) instantiate this general reduction using new and known learning algorithms to obtain new computationally efficient differentially private data release algorithms.

Before giving more details on our reduction in Section 1.1, we briefly discuss its context and some of the ways that we apply/instantiate it. While the search for efficient differentially private data release algorithms is relatively new, there are decades of work in learning theory aimed at developing techniques and algorithms for computationally efficient learning, going back to the early work of Valiant [Val84]. Given the high-level similarity between the two fields, leveraging the existing body of work and insights from learning theory for data release is a promising direction for future research; we view our reduction as a step in this direction. We note that our work is by no means the first to draw a connection between privacy-preserving data release and learning theory; as discussed in the "Related Work" section below, several prior works used learning techniques in the data release setting. A novelty in our work is that it gives an explicit and modular reduction from data release to natural learning problems. Conceptually, our reduction overcomes two main hurdles:

- bridging the gap between the *noisy* oracle access arising in private data release and the *noise-free* oracle access required by many learning algorithms (including the ones we use).

- avoiding any dependence on the database size in the complexity of the learning algorithm being used.

We use this reduction to construct new data release algorithms. In this work we explore two main applications of our reduction. The first aims to answer boolean conjunction queries (also known as contingency tables or marginal queries), one of the most well-motivated and widely-studied classes of statistical queries in the differential privacy literature. Taking the data universe $\mathcal{U}$ to be $\{0, 1\}^d$, the k-way boolean conjunction corresponding to a subset S of k attributes in $[d]$ counts what fraction of items in the database have all the attributes in S set to 1. Approximating the answers for k-way conjunctions (or all conjunctions) has been the focus of several past works (see, e.g. [BCD$^+$07, KRSU10, UV11, GHRU11]). Applying our reduction with a new learning algorithm tailored for this class, we obtain a data release algorithm that, for databases of size $d^{O\left(\sqrt{k \log(k \log d)}\right)}$, releases accurate answers to all k-way conjunctions simultaneously (we ignore for now the dependence of the database size on other parameters such as the error). The running time is $\mathrm{poly}(d^k)$. Previous algorithms either had running time $2^{\Omega(d)}$ (e.g. [DNR$^+$09]) or required a database of size $d^{k/2}$ (adding independent noise [DMNS06]). We also obtain better bounds for the task of approximating the answers to a large fraction of *all* (i.e. d-way) conjunctions under arbitrary distributions. These results follow from algorithms for learning thresholds of sums of the relevant predicates; we base these algorithms on learning theory techniques for representing such functions as low-degree polynomial threshold functions, following works such as [KS04, KOS04]. We give an overview of these results in Section 1.2 below.

Our second application uses Fourier analysis of the database (viewed, again, as a real-valued function on the queries in $\mathcal{Q}$). We obtain a new quasi-polynomial data release algorithm for low-depth (AC^0) counting queries. This uses an algorithm for learning Majority-of-AC^0 circuits due to Jackson *et al.* [JKS02]. We elaborate on this result in Section 1.3 below.

1.1 Private Data Release Reduces to Learning Thresholds In this section we give more details on the reduction from privacy-preserving data release to learning thresholds. The full details are in Sections 3 and 4. We begin with loose definitions of the data release and learning tasks we consider, and then proceed with (a simple case of) our reduction.

Counting Queries, Data Release and Learning Thresholds. We begin with preliminaries and an informal specification of the data release and learning tasks we consider in our reduction (see Sections 2 and 3.1 for full definitions). We refer to an element u in data domain $\mathcal{U}$ as an *item*. A *database* is a collection of n items from $\mathcal{U}$. A *counting query* is specified by a predicate $p : \mathcal{U} \to \{0,1\}$, and the query q_p on database D outputs the fraction of items in D that satisfy p, i.e. $\frac{1}{n}\sum_{i=1}^{n} p(D_i)$. A *class* of counting queries is specified by a set $\mathcal{Q}$ of query descriptions and a predicate $P \colon \mathcal{Q} \times \mathcal{U} \to \{0,1\}$. For a query $q \in \mathcal{Q}$, its corresponding predicate is $P(q,\cdot) : \mathcal{U} \to \{0,1\}$. We will sometimes fix a data item $u \in \mathcal{U}$ and consider the predicate $p_u(\cdot) \triangleq P(\cdot, u) : \mathcal{Q} \to \{0,1\}$.

Fix a data domain $\mathcal{U}$ and query class $\mathcal{Q}$ (specified by a predicate P). A *data release algorithm* $\mathcal{A}$ gets as input a database D, and access to a distribution G over $\mathcal{Q}$ (see below), and outputs a *synopsis* $S : \mathcal{Q} \to [0,1]$ that provides approximate answers to queries in $\mathcal{Q}$. We say that $\mathcal{A}$ is an (α,β,γ) *distribution-free data release algorithm for* $(\mathcal{U},\mathcal{Q},P)$ if, for any distribution G over the query set $\mathcal{Q}$, with probability $1-\beta$ over the algorithm's coins, the synopsis S satisfies that with probability $1-\gamma$ over $q \sim G$, the (additive) error of S on q is bounded by α. Later we will also consider data release algorithms that only work for a specific distribution or class of distributions (in this case we will not call the algorithm distribution-free). Finally, we assume for now that the data release algorithm only accesses the distribution G by sampling queries from it, but later we will also consider more general types of access (see below). A *differentially private* data release algorithm is one whose output distribution (on synopses) is differentially private as per Definition 1. See Definition 4 for full and formal details.

Fix a class $\mathcal{Q}$ of *examples* and a set $\mathcal{F}$ of predicates on $\mathcal{Q}$. Let $\mathcal{F}_{n,t}$ be the set of thresholded sums from $\mathcal{F}$, i.e., the set of functions of the form $f = \mathbb{I}\left\{\frac{1}{n}\sum_{i=1}^{n} f_i \geqslant t\right\}$, where $f_i \in \mathcal{F}$ for all $1 \leqslant i \leqslant n$. We refer to functions in $\mathcal{F}_{n,t}$ as *n-thresholds*. An algorithm for learning thresholds gets access to a function in $\mathcal{F}_{n,t}$ and outputs a *hypothesis* $h : \mathcal{Q} \to \{0,1\}$ that labels examples in $\mathcal{Q}$. We say that it is a (γ,β) distribution-free learning algorithm for learning thresholds over $(\mathcal{Q},\mathcal{F})$ if, for any distribution G over the set $\mathcal{Q}$, with probability $1-\beta$ over the algorithm's coins the output hypothesis h satisfies that with probability $1-\gamma$ over $q \sim G$, h labels q correctly. As above, later we will also consider learning algorithms that are not distribution free, and only work for a specific distribution or class or distributions. For now, we assume that the learning algorithm only accesses the distribution G by drawing examples from it. These examples are labeled using the target function that the algorithm is trying to learn. See Definition 5 for full and formal details.

The Reduction. We can now describe (a simple case of) our reduction from differentially private data release to learning thresholds. For any data domain $\mathcal{U}$, set $\mathcal{Q}$ of query descriptions, and predicate $P : \mathcal{Q} \times \mathcal{U} \to \{0,1\}$, the reduction shows how to construct a (distribution free) data release algorithm given a (distribution free) algorithm for learning thresholds over $(\mathcal{Q}, \{p_u \colon u \in \mathcal{U}\})$, i.e., any algorithm for learning thresholds where $\mathcal{Q}$ is the example set and the set $\mathcal{F}$ of predicates (over $\mathcal{Q}$) is obtained by the possible ways of fixing the u-input to P. The resulting data release algorithm is (α,β,γ)-accurate as long as the database is not too small; the size bound depends on the desired accuracy parameters and on the learning algorithm's sample complexity. The efficiency of the learning algorithm is preserved (up to mild polynomial factors).

Theorem 1.1. (Reduction from Data Release to Learning Thresholds, Simplified) *Let $\mathcal{U}$ be a data universe, $\mathcal{Q}$ a set of query descriptions, and $P\colon \mathcal{Q}\times \mathcal{U} \to \{0,1\}$ a predicate. There is an ε-differentially private (α,β,γ)-accurate distribution free data-release algorithm for $(\mathcal{U},\mathcal{Q},P)$, provided that:*

1. *there is a distribution-free learning algorithm $\mathcal{L}$ that (γ,β)-learns thresholds over $(\mathcal{Q},\{p_u\colon u\in\mathcal{U}\})$ using $b(n,\gamma,\beta)$ labeled examples and running time $t(n,\gamma,\beta)$ for learning n-thresholds.*
2. *$n \geqslant \frac{C\cdot b(n',\gamma',\beta')\cdot\log(1/\beta)}{\varepsilon\cdot\alpha\cdot\gamma}$, where $n'=\Theta(\log|\mathcal{Q}|/\alpha^2)$, $\beta'=\Theta(\beta\cdot\alpha)$, $\gamma'=\Theta(\gamma\cdot\alpha)$, $C=\Theta(1)$.*

Moreover, the data release algorithm only accesses the query distribution by sampling. The number of samples taken is $O(b(n',\gamma',\beta')\cdot\log(1/\beta)/\gamma)$ and the running time is $\mathrm{poly}(t(n',\gamma',\beta'),n,1/\alpha,\log(1/\beta),1/\gamma)$.

Section 3.2 gives a formal (and more general) statement in Theorem 3.1. Section 3.3 gives a proof overview, and Section 4 gives the full proof. Note that, since the data release algorithm we obtain from this reduction is distribution free (i.e. works for any distribution on the query set) and only accesses the query distribution by sampling, it can be *boosted* to yield accurate answers on *all* the queries [DRV10].

A More General Reduction. For clarity of exposition, we gave above a simplified form of the reduction. This assumed that the learning algorithm is *distribution-free* (i.e. works for any distribution over examples) and only requires sampling access to labeled examples. These strong assumptions enable us to get a distribution-free data release algorithm that only accesses the query distribution by sampling.

We also give a reduction that applies even to distribution-specific learning algorithms that require (a certain kind of) oracle access to the function being learned. In addition to sampling labeled examples, the learning algorithm can: (*i*) estimate the distribution G on any example q by querying q and receiving a (multiplicative) approximation to the probability $G[q]$; and (*ii*) query an oracle for the function f being learned on any q such that $G[q]\neq 0$. We refer to this as *approximate distribution restricted oracle access*, see Definition 6. Note that several natural learning algorithms in the literature use oracle queries in this way; in particular, we show that this is true for Jackson's Harmonic Sieve Algorithm [Jac97], see Section 6.

Our general reduction gives a data release algorithm for a class $\mathcal{GQ}$ of distributions on the query set, provided we have a learning algorithm which can also use approximate distribution restricted oracle access, and which works for a slightly richer class of distributions $\mathcal{GQ}'$ (a *smooth extension*, see Definition 7). Again, several such algorithms (based on Fourier analysis) are known in the literature; our general reduction allows us to use them and obtain the new data release results outlined in Section 1.3.

Related Work: Privacy and Learning. Our new reduction adds to the fruitful and growing interaction between the fields of differentially private data release and learning theory. Prior works also explored this connection. In our work, we "import" learning theory techniques by drawing a correspondence between the database (in the data release setting), for which we want to approximate query answers, and the target function (in the learning setting) which labels examples. Several other works have used this correspondence (implicitly or explicitly), e.g. [DNR$^+$09, DRV10, GHRU11]. A different view, in which *queries* in the data release setting correspond to *concepts* in learning theory, was used in [BLR08] and also in [GHRU11].

There is also work on *differentially private learning algorithms* in which the goal is to give differentially private variants of various learning algorithms [BDMN05, KLN$^+$08].

1.2 Applications (Part I): Releasing Conjunctions

We use the reduction of Theorem 1.1 to obtain new data release algorithms "automatically" from learning algorithms that satisfy the theorem's requirements. Here we describe the distribution-free data release algorithms we obtain for approximating conjunction counting queries. These use learning algorithms (which are themselves distribution-free and require only random examples) based on polynomial threshold functions.

Throughout this section we fix the query class under consideration to be conjunctions. We take $\mathcal{U}=\{0,1\}^d$, and a (monotone) conjunction $q\in\mathcal{Q}=\{0,1\}^d$ is satisfied by u iff $\forall i$ s.t. $q_i=1$ it is also the case that $u_i=1$. (Our monotone conjunction results extend easily to general non-monotone conjunctions with parameters unchanged.[1]) Our first result is an algorithm for releasing k-way conjunctions:

Theorem 1.2. (Distribution-Free Data Release for k-way conjunctions) *There is an ε-differentially*

[1] To see this, extend the data domain to be $\{0,1\}^{2d}$, and for each item in the original domain include also its negation. General conjunctions in the original data domain can now be treated as monotone conjunctions in the new data domain. Note that the locality of a conjunction is unchanged. Our results in this section are for arbitrary distributions over the set of monotone conjunctions (over the new domain), and so they will continue to apply to arbitrary distributions on general conjunctions over the original data domain.

private (α, β, γ)-accurate distribution-free data release algorithm, which accesses the query distribution only by sampling, for the class of k-way monotone Boolean conjunction queries. The algorithm has runtime $\mathrm{poly}(n)$ *on databases of size n provided that*

$$n \geqslant d^{O\left(\sqrt{k \log\left(\frac{k \log d}{\alpha}\right)}\right)} \cdot \tilde{O}\left(\frac{\log\left(1/\beta\right)^3}{\varepsilon\alpha\gamma^2}\right).$$

Since this is a distribution-free data release algorithm that only accesses the query distribution by sampling, we can use the boosting results of [DRV10] and obtain a data release algorithm that generates (w.h.p.) a synopsis that is accurate for *all* queries. This increases the running time to $d^k \cdot \mathrm{poly}(n)$ (because the boosting algorithm needs to enumerate over all the k-way conjunctions). The required bound on the database size increases slightly but our big-Oh notation hides this small increase. The corollary is stated formally below:

Corollary 1.1. (Boosted Data Release for k-way Conjunctions) *There is an ε-differentially private $(\alpha, \beta, \gamma = 0)$-accurate distribution-free data release algorithm for the class of k-way monotone Boolean conjunction queries with runtime $d^k \cdot \mathrm{poly}(n)$ on databases of size n, provided that*

$$n \geqslant d^{O\left(\sqrt{k \log\left(\frac{k \log d}{\alpha}\right)}\right)} \cdot \tilde{O}\left(\frac{\log\left(1/\beta\right)^3}{\varepsilon\alpha}\right).$$

We also obtain a new data release algorithm for releasing the answers to *all* conjunctions:

Theorem 1.3. (Distribution-Free Data Release for *All* Conjunctions) *There is an ε-differentially private (α, β, γ)-accurate distribution-free data release algorithm, which accesses the query distribution only by sampling, for the class of all monotone Boolean conjunction queries. The algorithm has runtime* $\mathrm{poly}(n)$ *on databases of size n, provided that*

$$n \geqslant d^{O\left(d^{1/3} \cdot \log^{2/3}\left(\frac{d}{\alpha}\right)\right)} \cdot \tilde{O}\left(\frac{\log\left(1/\beta\right)^3}{\varepsilon\alpha\gamma^2}\right).$$

Again, we can apply boosting to this result; this gives improvements over previous work for a certain range of parameters (roughly $k \in [d^{1/3}, d^{2/3}]$). We omit the details.

Related Work on Releasing Conjunctions. Several past works have considered differentially private data release for conjunctions and k-way conjunctions (also known as marginals and contingency tables). As a corollary of their more general Laplace and Gaussian mechanisms, the work of Dwork *et al.* [DMNS06] showed how to release all k-way conjunctions in running time $d^{O(k)}$ provided that the database size is at least $d^{O(k)}$. Barak *et al.* [BCD$^+$07] showed how to release *consistent* contingency tables with similar database size bounds. The running time, however, was increased to $\exp(d)$. We note that our data-release algorithms do not guarantee consistency. Gupta *et al.* gave *distribution-specific* data release algorithm for k-way and for all conjunctions. These algorithms work for the uniform distribution over (k-way or general) conjunctions. The database size bound and running time were (roughly) $d^{\tilde{O}(1/\alpha^2)}$. For distribution-specific data release on the uniform distribution, the dependence on d in their work is better than our algorithms but the dependence on α is worse. Finally, we note that the general information-theoretic algorithms for differentially private data release also yield algorithms for the specific case of conjunctions. These algorithms are (significantly) more computationally expensive, but they have better database size bounds. For example, the algorithm of [HR10] has running time $\exp(d)$ but database size bound is (roughly) $\tilde{O}(d/\alpha^2)$ (for the relaxed notion of (ε, δ)-differential privacy).

In terms of negative results, Ullman and Vadhan [UV11] showed that, under mild cryptographic assumptions, no data release algorithm for conjunctions (even 2-way) can output a *synthetic database* in running time less than $\exp(d)$ (this holds even for *distribution-specific* data release on the uniform distribution). Our results side-step this negative result because the algorithms do not release a synthetic database.

Kasiviswanathan *et al.* [KRSU10] showed a lower bound of $\tilde{\Omega}\left(\min\left\{d^{k/2}/\alpha, 1/\alpha^2\right\}\right)$ on the database size needed for releasing k-way conjunctions. To see that this is consistent with our bounds, note that our bound on n is always larger than $f(\alpha) = 2^{\sqrt{k \log(1/\alpha)}}/\alpha$. We have $f(\alpha) < 1/\alpha^2$ only if $k < \log(1/\alpha)$. But in the range where $k < \log(1/\alpha)$ our theorem needs n to be larger than d^k/α which is consistent with the lower bound.

1.3 Applications (Part II): AC^0-Counting Queries

We also use Theorem 1.1 (in its more general formulation given in Section 3.2) to obtain a new data release algorithm for answering general AC^0 counting queries (in quasi-polynomial time). Here we fix the data universe to be $\mathcal{U} = \{0,1\}^d$, and take the set of query descriptions to also be $\mathcal{Q} = \{0,1\}^d$. The algorithm is distribution-specific, working for the uniform distribution over query descriptions,[2] and instantiates the reduction with a learning algorithm

[2] More generally, we can get results for *smooth* distributions, we defer these to the full version.

that uses Fourier analysis of the target function. Thus the full data release algorithm uses Fourier analysis of the database (viewed as a function on queries).

Consider any query family whose predicate is computed by a constant depth (AC^0) circuit. For any family of this type, in Section 6 we obtain a data release algorithm over the uniform distribution that requires a database of quasi-polynomial (in d) size (and has running time polynomial in the database size, or quasi-polynomial in d).

Theorem 1.4. (Uniform Distribution Data Release for AC^0 Counting Queries) *Take $\mathcal{U} = \mathcal{Q} = \{0,1\}^d$, and $P(q,u) : \mathcal{Q} \times \mathcal{U} \to \{0,1\}$ a predicate computed by a Boolean circuit of depth $\ell = O(1)$ and size* $\mathrm{poly}(d)$. *There is an ε-differentially private data release algorithm for this query class over the uniform distribution on $\mathcal{Q}$. For databases of size n, the algorithm has runtime* $\mathrm{poly}(n)$ *and is (α,β,γ)-accurate, provided that:*

$$n \geqslant d^{O\left(\log^{\ell}\left(\frac{d}{\alpha\gamma}\right)\right)} \cdot \tilde{O}\left(\frac{\log^3(1/\beta)}{\varepsilon\alpha^2\gamma}\right).$$

This result uses our reduction instantiated with an algorithm of Jackson *et al.* [JKS02] for learning Majority-of-AC^0 circuits. To the best of our knowledge, this is the first positive result for private data release that uses the (circuit) structure of the query class in a "non black-box" way to approximate the query answer. We note that the class of AC^0 predicates is quite rich. For example, it includes conjunctions, approximate counting [Ajt83], and $GF[2]$ polynomials with $\mathrm{polylog}(d)$ many terms. While our result is specific to the uniform distribution over $\mathcal{Q}$, we note that some query sets (and query descriptions) may be amenable to *random self-reducibility*, where an algorithm providing accurate answers to uniformly random $q \in Q$ can be used to get (w.h.p.) accurate answers to *any* $q \in Q$. We also note that Theorem 1.4 leaves a large degree of freedom in how a class of counting queries is to be represented. Many different sets of query descriptions $\mathcal{Q}$ and predicates $P(q,u)$ can correspond to the same set of counting queries over the same $\mathcal{U}$, and it may well be the case that some representations are more amenable to computations in AC^0 and/or random self-reducibility. Finally, we note that the hardness results of Dwork *et al.* [DNR$^+$09] actually considered (and ruled out) efficient data-release algorithms for AC^0 counting queries (even for the uniform distribution case), but only when the algorithm's output is a synthetic database. Theorem 1.4 side-steps these negative results because the output is not a synthetic database.

2 Preliminaries

Data sets and differential privacy. We consider a data universe $\mathcal{U}$, where throughout this work we take $\mathcal{U} = \{0,1\}^d$. We typically refer to an element $u \in \mathcal{U}$ as an *item*. A data set (or database) D of size n over the universe $\mathcal{U}$ is an ordered multiset consisting of n items from $\mathcal{U}$. We will sometimes think of D as a tuple in $\mathcal{U}^n$. We use the notation $|D|$ to denote the size of D (here, n). Two data sets D, D' are called *adjacent* if they are both of size n and they agree in at least $n-1$ items (i.e., their edit distance is at most 1).

We will be interested in randomized algorithms that map data sets into some abstract range $\mathcal{R}$ and satisfy the notion of differential privacy.

Definition 1. (Differential Privacy [DMNS06]) *A randomized algorithm $\mathcal{M}$ mapping data sets over $\mathcal{U}$ to outcomes in $\mathcal{R}$ satisfies* (ε,δ)-differential privacy *if for all $S \subset \mathcal{R}$ and every pair of two adjacent databases D, D', we have $\mathbb{P}(\mathcal{M}(D) \in S) \leqslant e^{\varepsilon}\,\mathbb{P}(\mathcal{M}(D') \in S) + \delta$. If $\delta = 0$, we say the algorithm satisfies* ε-differential privacy.

Counting queries. A class of *counting queries* is specified by a predicate $P\colon \mathcal{Q} \times \mathcal{U} \to \{0,1\}$ where $\mathcal{Q}$ is a set of query descriptions. Each $q \in \mathcal{Q}$ specifies a query and the answer for a query $q \in \mathcal{Q}$ on a single data item $u \in \mathcal{U}$ is given by $P(q,u)$. The answer of a counting query $q \in \mathcal{Q}$ on a data set D is defined as $\frac{1}{n}\sum_{u \in D} P(q,u)$.

We will often fix a data item u and database $D \in \mathcal{U}^n$ of n data items, and use the following notation:

- $p_u\colon \mathcal{Q} \to \{0,1\}$, $p_u(q) \stackrel{\text{def}}{=} P(q,u)$. The predicate on a fixed data item u.
- $f^D\colon \mathcal{Q} \to [0,1]$, $f^D(q) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{u \in D} P(q,u)$. For an input query description and fixed database, counts the fraction of database items that satisfy that query.
- $f^D_t\colon \mathcal{Q} \to \{0,1\}$, $f^D_t(q) \stackrel{\text{def}}{=} \mathbb{I}\left\{f^D(q) \geqslant t\right\}$. For an input query description and fixed database and threshold $t \in [0,1]$, indicates whether the fraction of database items that satisfy that query is at least t. Here and in the following $\mathbb{I}$ denotes the 0/1-indicator function.

We close this section with some concrete examples of query classes that we will consider. Fix $\mathcal{U} = \{0,1\}^d$ and $\mathcal{Q} = \{0,1\}^d$. The query class of *monotone boolean conjunctions* is defined by the predicate $P(q,u) = \bigwedge_{i\colon q_i = 1} u_i$. Note that we may equivalently write $P(q,u) = 1 - \bigvee_{i\colon u_i = 0} q_i$. The query class of *parities over* $\{0,1\}^d$ is defined by the predicate $P(q,u) = \sum_{i: u_i = 1} q_i \pmod 2$.

3 Private Data Release via Learning Thresholds

In this section we describe our reduction from private data release to a related computational learning task of learning thresholded sums. Section 3.1 sets the stage, first introducing definitions for handling distributions and access to an oracle, and then proceeds with notation and formal definitions of (non-interactive) data release and of learning threshold functions. Section 3.2 formally states our main theorem giving the reduction, and Section 3.3 gives an intuitive overview of the proof. The formal proof is then given in Section 4.

3.1 Distribution access, data release, learning thresholds

Definition 2. (Sampling or Evaluation Access to a Distribution) *Let G be a distribution over a set $\mathcal{Q}$. When we give an algorithm $\mathcal{A}$* sampling access *to G, we mean that $\mathcal{A}$ is allowed to sample items distributed by G. When we give an algorithm $\mathcal{A}$* evaluation access *to G, we mean that $\mathcal{A}$ is both allowed to sample items distributed by G and also to make oracle queries: in such a query $\mathcal{A}$ specifies any $q \in Q$ and receives back the probability $G[q] \in [0,1]$ of q under G. For both types of access we will often measure $\mathcal{A}$'s* sample complexity *or* number of queries *(for the case of evaluation access).*[3]

Definition 3. (Sampling Access to Labeled Examples) *Let G be a distribution over a set $\mathcal{Q}$ of potential examples, and let f be a function whose domain is $\mathcal{Q}$. When we give an algorithm $\mathcal{A}$* sampling access to labeled examples by (G, f), *we mean that $\mathcal{A}$ has sampling access to the distribution $(q, f(q))_{q \sim G}$.*

Definition 4. (Data Release Algorithm) *Fix $\mathcal{U}$ to be a data universe, $\mathcal{Q}$ to be a set of query descriptions, $\mathcal{GQ}$ to be a set of distributions on $\mathcal{Q}$, and $P(q,u) : \mathcal{Q} \times \mathcal{U} \to \{0,1\}$ to be a predicate. A $(\mathcal{U}, \mathcal{Q}, \mathcal{GQ}, P)$* data release algorithm *$\mathcal{A}$ is a (probabilistic) algorithm that gets sampling access to a distribution $G \in \mathcal{GQ}$ and takes as input accuracy parameters $\alpha, \beta, \gamma > 0$, a database size n, and a database $D \in \mathcal{U}^n$. $\mathcal{A}$ outputs a* synopsis $S : \mathcal{Q} \to [0,1]$.

We say that $\mathcal{A}$ is (α, β, γ)-accurate *for databases of size n, if for every database $D \in \mathcal{U}^n$ and query distribution $G \in \mathcal{GQ}$:*

(3.1)
$$\mathop{\mathbb{P}}_{S \leftarrow \mathcal{A}(n,D,\alpha,\beta,\gamma)} \left[\mathop{\mathbb{P}}_{q \sim G} \left[|S(q) - f^D(q)| > \alpha \right] > \gamma \right] < \beta$$

We also consider data release algorithms that get evaluation access to G. In this case, we say that $\mathcal{A}$ is a data release algorithm using evaluation access. *The definition is unchanged, except that $\mathcal{A}$ gets this additional form of access to G.*

When P and $\mathcal{U}$ are understood from the context, we sometimes refer to a $(\mathcal{U}, \mathcal{Q}, \mathcal{GQ}, P)$ data release algorithm as an *algorithm for releasing the class of queries $\mathcal{Q}$ over $\mathcal{GQ}$.*

This work focuses on *differentially private* data release algorithms, i.e. data release algorithms which are ε-differentially private as per Definition 1 (note that such algorithms must be randomized). In such data release algorithms, the probability of any output synopsis S differs by at most an e^{ε} multiplicative factor between any two adjacent databases.

We note two cases of particular interest. The first is when $\mathcal{GQ}$ is the set of *all distributions* over $\mathcal{Q}$. In this case, we say that $\mathcal{A}$ is a *distribution-free* data release algorithm. For such algorithms it is possible to apply the "boosting for queries" results of [DRV10] and obtain a data release algorithm whose synopsis is (w.h.p.) accurate on *all* queries (i.e. with $\gamma = 0$). We note that those boosting results apply only to data release algorithms that access their distribution by sampling (i.e. they need not hold for data release algorithms that use evaluation access).

A second case of interest is when $\mathcal{GQ}$ contains only a single distribution, the uniform distribution over all queries $\mathcal{Q}$. In this case both sampling and evaluation access are easy to simulate.

Remark 1. *Throughout this work, we fix the accuracy parameter α, and lower-bound the required database size n needed to ensure the (additive) approximation error is at most α. An alternative approach taken in some of the differential privacy literature, is fixing the database size n and upper bounding the approximation error α as a function of n (and of the other parameters). Our database size bounds can be converted to error bounds in the natural way.*

Definition 5. (Learning Thresholds) *Let $\mathcal{Q}$ be a set (which we now view as a domain of potential unlabeled examples) and let $\mathcal{GQ}$ be a set of distributions on $\mathcal{Q}$. Let $\mathcal{F}$ be a set of predicates on $\mathcal{Q}$, i.e. functions $Q \to \{0,1\}$. Given $t \in [0,1]$, let $\mathcal{F}_{n,t}$ be the set of all threshold functions of the form $f = \mathbb{I}\left\{\frac{1}{n}\sum_{i=1}^{n} f_i \geqslant t\right\}$ where $f_i \in \mathcal{F}$ for all $1 \leqslant i \leqslant n$. We refer to functions in $\mathcal{F}_{n,t}$ as* n-thresholds *over $\mathcal{F}$. Let $\mathcal{L}$ be a (probabilistic) algorithm that gets sampling access to labeled examples by a distribution $G \in \mathcal{GQ}$ and a target function $f \in \mathcal{F}_{n,t}$. $\mathcal{L}$ takes as input accuracy parameters $\gamma, \beta > 0$, an in-*

[3] Note that, generally speaking, sampling and evaluation access are incomparably powerful (see [KMR+94, Nao96]). In this work, however, whenever we give an algorithm evaluation access we will also give it sampling access.

teger $n > 0$, and a threshold $t \in [0,1]$. $\mathcal{L}$ outputs a boolean hypothesis $h : \mathcal{Q} \to \{0,1\}$.

We say that $\mathcal{L}$ is an (γ, β)-learning algorithm for thresholds over $(\mathcal{Q}, \mathcal{GQ}, \mathcal{F})$ *if for every $\gamma, \beta > 0$, every n, every $t \in [0,1]$, every $f \in \mathcal{F}_{n,t}$ and every $G \in \mathcal{GQ}$, we have*

$$\underset{h \leftarrow \mathcal{L}(n,t,\gamma,\beta)}{\mathbb{P}} \left[\underset{q \sim G}{\mathbb{P}} [h(q) \neq f(q)] > \gamma \right] < \beta \,. \tag{3.2}$$

The definition is analogous for all other notions of oracle access (see e.g. Definition 6 below).

3.2 Statement of the main theorem

In this section we formally state our main theorem, which establishes a general reduction from private data release to learning certain threshold functions. The next definition captures a notion of oracle access for learning algorithms which arises in the reduction. The definition combines sampling access to labeled examples with a limited kind of evaluation access to the underlying distribution and black-box oracle access to the target function f.

Definition 6. (approximate distribution-restricted oracle access) *Let G be a distribution over a domain $\mathcal{Q}$, and let f be a function whose domain is $\mathcal{Q}$. When we say that an algorithm $\mathcal{A}$ has* approximate G-restricted evaluation access to f, *we mean that*

1. *$\mathcal{A}$ has sampling access to labeled examples by (G, f); and*
2. *$\mathcal{A}$ can make oracle queries on any $q \in Q$, which are answered as follows: there is a fixed constant $c \in [1/3, 3]$ such that (i) if $G[q] = 0$ the answer is $(0, \perp)$; and (ii) if $G[q] > 0$ the answer is a pair $(c \cdot G[q], f(q))$.*

Remark 2. *We remark that this is the type of of oracle access provided to the learning algorithm in our reduction. This is different from the oracle access that the data release algorithm has. We could extend Definition 4 to refer to approximate evaluation access to G; all our results on data release using evaluation access would extend to this weaker access (under appropriate approximation guarantees). For simplicity, we focus on the case where the* data release algorithm *has perfectly accurate evaluation access, since this is sufficient throughout for our purpose.*

One might initially hope that privately releasing a class of queries $\mathcal{Q}$ over some set of distributions $\mathcal{GQ}$ reduces to learning corresponding threshold functions over the *same* set of distributions. However, our reduction will need a learning algorithm that works for a potentially larger set of distributions $\mathcal{GQ}' \supseteq \mathcal{GQ}$. (We will see in Theorem 3.1 that this poses a stronger requirement on the learning algorithm.) Specifically, $\mathcal{GQ}'$ will be a *smooth extension* of $\mathcal{GQ}$ as defined next.

Definition 7. (smooth extensions) *Given a distribution G over a set $\mathcal{Q}$ and a value $\mu \geqslant 1$, the μ*-smooth extension of G *is the set of all distributions G' which are such that $G'[q] \leqslant \mu \cdot G[q]$ for all $q \in \mathcal{Q}$. Given a set of distributions $\mathcal{GQ}$ and $\mu \geqslant 1$, the μ*-smooth extension of $\mathcal{GQ}$, *denoted $\mathcal{GQ}'$, is defined as the set of all distributions that are a μ-smooth extension of some $G \in \mathcal{GQ}$.*

With these two definitions at hand, we can state our reduction in its most general form. We will combine this general reduction with specific learning results to obtain concrete new data release algorithms in Sections 5 and 6.

Theorem 3.1. (Main Result: Private Data Release via Learning Thresholds) *Let $\mathcal{U}$ be a data universe, $\mathcal{Q}$ a set of query descriptions, $\mathcal{GQ}$ a set of distributions over $\mathcal{Q}$, and $P \colon \mathcal{Q} \times \mathcal{U} \to \{0,1\}$ a predicate.*

Then, there is an ε-differentially private (α, β, γ)-accurate data-release algorithm for databases of size n provided that

- *there is an algorithm $\mathcal{L}$ that (γ,β)-learns thresholds over $(\mathcal{Q}, \mathcal{GQ}', \{p_u \colon u \in \mathcal{U}\})$, running in time $t(n, \gamma, \beta)$ and using $b(n, \gamma, \beta)$ queries to an approximate distribution-restricted evaluation oracle for the target n-threshold function, where $\mathcal{GQ}'$ is the $(2/\gamma)$-smooth extension of $\mathcal{GQ}$; and*
- *we have*

$$n \geqslant \frac{C \cdot b(n', \gamma', \beta') \cdot \log\left(\frac{b(n', \gamma', \beta')}{\alpha\gamma\beta}\right) \cdot \log(1/\beta')}{\varepsilon\alpha^2\gamma} \,, \tag{3.3}$$

  where $n' = \Theta(\log|\mathcal{Q}|/\alpha^2)$, $\beta' = \Theta(\beta\alpha)$, $\gamma' = \Theta(\gamma\alpha)$ and $C > 0$ is a sufficiently large constant.

The running time of the data release algorithm is $\mathrm{poly}(t(n', \gamma', \beta'), n, 1/\alpha, \log(1/\beta), 1/\gamma)$.

The next remark points out two simple modifications of this theorem.

Remark 3. 1. *We can improve the dependence on n in (3.3) by a factor of $\Theta(1/\alpha)$ in the case where the learning algorithm $\mathcal{L}$ only uses sampling access to labeled examples. In this case the data release algorithm also uses only sampling access to the query distribution G. The precise statement is given in Theorem 4.1 which we present after the proof of Theorem 3.1.*

2. *A similar theorem holds for (ε, δ)-differential privacy, where the requirement on n in (3.3) is improved to a requirement on $\sqrt{n}$ up to a $\log(1/\delta)$ factor. The proof is the same, except for a different (but standard) privacy argument, e.g., using the Composition Theorem in [DRV10].*

3.3 Informal proof overview Our goal in the data release setting is approximating the query answers $\{f^D(q)\}_{q \in Q}$. This is exactly the task of approximating or *learning* a sum of n predicates from the set $\mathcal{F} = \{p_u \colon u \in \mathcal{U}\}$. Indeed, each item u in the database specifies a predicate p_u, and for a fixed query $q \in Q$ we are trying to approximate the sum of the predicates $f^D(q) = \frac{1}{|D|} \cdot \sum_{u \in D} p_u(q)$. We want to approximate such a sum in a privacy-preserving manner, and so we will only permit limited access to the function f^D that we try to approximate. In particular, we will only allow a bounded number of noisy oracle queries to this function. Using standard techniques (i.e. adding appropriately scaled Laplace noise [DMNS06]), an approximation obtained from a bounded number of noisy oracle queries will be differentially private. It remains, then, to tackle the task of (i) learning a sum of n predicates from $\mathcal{F}$ using an oracle to the sum, and (ii) doing so using only a *bounded (smaller than n) number* of oracle queries when we are provided *noisy answers.*

From Sums to Thresholds. Ignoring privacy concerns, it is straightforward to reduce the task of learning a sum f^D of predicates (given an oracle for f^D) to the task of learning thresholded sums of predicates (again given an oracle for f^D). Indeed, set $k = \lceil 3/\alpha \rceil$ and consider the thresholds $t_1, \ldots, t_k$ given by $t_i = i/(k+1)$. Now, given an oracle for f^D, it is easy to simulate an oracle for $f^D_{t_i}$ for any t_i. Thus, we can learn each of the threshold functions $f^D_{t_i}$ to accuracy $1 - \gamma/k$ with respect to G. Call the resulting hypotheses $h_1, \ldots, h_k$. Each h_i labels a $(1-\gamma/k)$-fraction of the queries/examples in Q correctly w.r.t the threshold function $f^D_{t_i}$. We can produce an aggregated hypothesis h for approximating f^D as follows: given a query/example q, let $h(q)$ equal t_i where t_i is the smallest i such that $h_i(q) = 0$ and $h_{i+1}(q) = 1$. For random $q \sim G$, we will then have $|h(q) - f^D(q)| \leqslant \alpha/3$ with probability $1 - \gamma$ (over the choice of q).

Thus, we have reduced learning a sum to learning thresholded sums (where in both cases the learning is done with an oracle for the sum). But because of privacy considerations, we must address the challenges mentioned above: (i) learning a *thresholded* sum of n predicates using few (less than n) oracle queries to the sum, and (ii) learning when the oracle for the sum can return noisy answers. In particular, the noisy sum answers can induce errors on threshold oracle queries (when the sum is close to the threshold).

Restricting to Large Margins. Let us say that a query/example $q \in \mathcal{Q}$ has *low margin with respect to* f^D *and* t_i if $|f^D(q) - t_i| \leqslant \alpha/7$. A useful observation is that in the argument sketched above, we do *not* need to approximate each threshold function $f^D_{t_i}$ well on low margin elements q. Indeed, suppose that each hypothesis h_i errs arbitrarily on a set $E_i \subseteq \mathcal{Q}$ that contains only inputs that have low margin w.r.t. f^D and t_i, but achieves high accuracy $1 - \gamma/k$ with respect to G conditioned on the event $\mathcal{Q} \setminus E_i$. Then the above aggregated hypothesis h would still have high accuracy with high probability over $q \sim G$; more precisely, h would satisfy $|h(q) - f^D(q)| \leqslant 2\alpha/3$ with probability $1-\gamma$ for $q \sim G$.

The reason is that for every $q \in Q$, there can only be one threshold $i^* \in \{1, \ldots, k\}$ such that $|f^D(q) - t_{i^*}| \leqslant \alpha/7$ (since any two thresholds are $\alpha/3$- apart from each other). While the threshold hypothesis h_{i^*} might err on q (because q has low margin w.r.t. t_{i^*}), the hypotheses h_{i^*-1} and h_{i^*+1} should still be accurate (w.h.p. over $q \sim G$), and thus the aggregated hypothesis h will still output a value between t_{i^*-1} and t_{i^*+1}.

Threshold Access to The Data Set. We will use the above observation to our advantage. Specifically, we restrict all access to the function f^D to what we call a *threshold oracle.* Roughly speaking, the threshold oracle (which we denote $\mathcal{TO}$ and define formally in Section 4.1) works as follows: when given a query q and a threshold t, it draws a suitably scaled Laplacian variable N (used to ensure differential privacy) and returns 1 if $f^D(q) + N \geqslant t + \alpha/20$; returns 0 if $f^D(q) + N \leqslant t - \alpha/20$; and returns "$\perp$" if $t - \alpha/20 < f^D(q) + N < t + \alpha/20$. If D is large enough then we can ensure that $|N| \leqslant \alpha/40$ with high probability, and thus whenever the oracle outputs $\perp$ on a query q we know that q has low margin with respect to f^D and t (since $\alpha/20 + |N| < \alpha/7$).

We will run the learning algorithm $\mathcal{L}$ on examples generated using the oracle $\mathcal{TO}$ after removing all examples for which the oracle returned $\perp$. Since we are conditioning on the $\mathcal{TO}$ oracle not returning $\perp$, this transforms the distribution G into a conditional distribution which we denote G'. Since we have only conditioned on removing low-margin q's, the argument sketched above applies. That is, the hypothesis that has high accuracy with respect to this conditional distribution G' is still useful for us.

So the threshold oracle lets us use noisy sum answers (allowing the addition of noise and differential privacy), but in fact it also addresses the second challenge of reducing the query complexity of the learning algorithm. This is described next.

Savings in Query Complexity via Subsampling. The remaining challenge is that the threshold oracle can be invoked only (at most) n times before we exceed our "privacy budget". This is problematic, because the query complexity of the underlying learning algorithm may well depend on n, since f^D is a sum of n predicates. To reduce the number of oracle queries that need to be made, we observe that the sum of n predicates that we are trying to learn can actually be approximated by a sum of fewer predicates. In fact, there exists a sum $f^{D'}$ of $n' = O(\log|\mathcal{Q}|/\alpha^2)$ predicates from $\mathcal{F}$ that is $\alpha/100$-close to f^D on all inputs in $\mathcal{Q}$, i.e. $|f^D(q) - f^{D'}(q)| \leqslant \alpha/100$ for all $q \in \mathcal{Q}$. (The proof is by a subsampling argument, as in [BLR08]; see Section 4.1.) We will aim to learn this "smaller" sum. The hope is that the query complexity for learning $f^{D'}$ may be considerably smaller, namely scaling with n' rather than n. Notice, however, that learning a threshold of $f^{D'}$ requires a threshold oracle to $f^{D'}$, rather than the threshold oracle we have, which is to f^D. Our goal, then, is to use the threshold oracle to f^D to simulate a threshold oracle to $f^{D'}$. This will give us "the best of both worlds": we can make (roughly) $O(n)$ oracle queries thus preserving differential privacy, while using a learning algorithm that is allowed to have query complexity superlinear in n'.

The key observation showing that this is indeed possible is that the threshold oracle $\mathcal{TO}$ already "avoids" low-margin queries where f_t^D and $f_t^{D'}$ might disagree! Whenever the threshold oracle $\mathcal{TO}$ (w.r.t. D) answers $l \neq \perp$ on a query q,, we must have $|f^D(q) - t| \geqslant \alpha/20 - N > \alpha/100$, and thus $f_t^D(q) = f_t^{D'}(q)$. Moreover, it is still the case that $\mathcal{TO}$ only answers $\perp$ on queries q that have low margins w.r.t $f_t^{D'}$. This means that, as above, we can run $\mathcal{L}$ using $\mathcal{TO}$ (w.r.t. D) in order to learn $f^{D'}$. The query complexity depends on n' and is therefore independent of n. At the same time, we continue to answer all queries using the threshold oracle with respect to f^D so that our privacy budget remains on the order $|D| = n$. Denoting the query complexity of the learning algorithm by $b(n')$ we only need that $n \gg b(n')$. This allows us to use learning algorithms that have $b(n') \gg n'$ as is usually the case.

Sampling from the conditional distribution. In the exposition above we glossed over one technical detail, which is that the learning algorithm requires sampling (or distribution restricted) access to the distribution G' over queries q on which $\mathcal{TO}$ does not return $\perp$, whereas the data release algorithm we are trying to build only has access to the original distribution G. We reconcile this disparity as follows.

For a threshold t, let ζ_t denote the probability that the oracle $\mathcal{TO}$ does not return $\perp$ when given a random $q \sim G$ and the threshold t. There are two cases depending on ζ_t:

$\zeta_t < \gamma$: This means that the threshold t is such that with probability $1-\gamma$ a random sample $q \sim G$ has low margin with respect to f^D and t. In this case, by simply outputting the constant-t function as our approximation for f^D, we get a hypothesis that has accuracy $\alpha/3$ with probability $1-\gamma$ over random $q \sim G$.

$\zeta_t \geqslant \gamma$: In this case, the conditional distribution G' induced by the threshold oracle is $1/\gamma$-smooth w.r.t. G. In particular, G' is contained in the smooth extension $\mathcal{GQ}'$ for which the learning algorithm is guaranteed to work (by the conditions of Theorem 3.1). This means that it we can sample from G' using rejection sampling to G. It suffices to oversample by a factor of $O(1/\gamma)$ to make sure that we receive enough examples that are not rejected by the threshold oracle.

Finally using a reasonably accurate estimate of ζ, we can also implement the distribution restricted approximate oracle access that may be required by the learning algorithm. We omit the details from this informal overview.

4 Proof of Theorem 3.1

In this section, we give a formal proof of Theorem 3.1. We formalize and analyze the threshold oracle first. Then we proceed to our main reduction.

4.1 Threshold access and subsampling

We begin by describing the threshold oracle that we use to access the function f^D throughout our reduction; it is presented in Figure 1. The oracle has two purposes. One is to ensure differential privacy by adding noise every time we access f^D. The other purpose is to "filter out" queries that are too close to the given threshold. This will enable us to argue that the threshold oracle for f_t^D agrees with the function $f_t^{D'}$ where D' is a small subsample of D.

Throughout the remainder of this section we fix all input parameters to our oracle, i.e. the data set D and the values $b, \alpha > 0$. We let $\beta > 0$ denote the desired error probability of our algorithm.

Lemma 4.1. *Call two queries $(q,t),(q',t')$* distinct *if $q \neq q'$. Then, the threshold oracle $\mathcal{TO}(D,\alpha,b)$ answers any sequence of b distinct adaptive queries to f^D with ε-differential privacy.*

Proof. This follows directly from the guarantees of the Laplacian mechanism as shown in [DMNS06]. □

Input: data set D of size n, tolerance $\alpha > 0$, query bound $b \in \mathbb{N}$.
Threshold Oracle $\mathcal{TO}(D, \alpha, b)$:

- When invoked on the j-th query $(q, t) \in \mathcal{Q} \times [0, 1)$, do the following:
  - If $j > b$, output $\perp$ and terminate.
  - If (q, t') has not been asked before for any threshold t', sample a fresh Laplacian variable $N_q \sim \mathrm{Lap}(b/\varepsilon n)$ and put $A_q = f^D(q) + N_q$. Otherwise reuse the previously created value A_q.
  - $$\text{Output} \begin{cases} 0 & \text{if } A_q \leqslant t - 2\alpha/3, \\ 1 & \text{if } A_q \geqslant t + 2\alpha/3, \\ \perp & \text{otherwise.} \end{cases}$$

Figure 1: Threshold oracle for f^D. This threshold oracle is the only way in which the data release algorithm ever interacts with the data set D. Its purpose is to ensure privacy and to reject queries that are too close to a given threshold.

Our goal is to use the threshold oracle for f_t^D to correctly answer queries to the function $f_t^{D'}$ where D' is a smaller (sub-sampled) database that gives "close" answers to D on all queries $q \in \mathcal{Q}$. The next lemma shows that there always exists such a smaller database.

Lemma 4.2. *For any $\alpha \geqslant 0$, there is a database D' of size*

$$|D'| \leqslant \frac{10 \log |\mathcal{Q}|}{\alpha^2} \tag{4.4}$$

such that

$$\max_{q \in \mathcal{Q}} \left| f^D(q) - f^{D'}(q) \right| < \alpha \, .$$

Proof. The existence of D' follows from a subsampling argument as shown in [BLR08]. □

The next lemma states the two main properties of the threshold oracle that we need. To state them more succinctly, let us denote by

$$Q(t, \alpha) = \{q \in \mathcal{Q} \colon |f^D(q) - t| \geqslant \alpha\}$$

the set of elements in $\mathcal{Q}$ that are α-far from the threshold t.

Lemma 4.3. (Agreement) *Suppose D satisfies*

$$|D| \geqslant \frac{30b \cdot \log(b/\beta)}{\varepsilon\alpha}, \tag{4.5}$$

Then, there is a data set D' of size $|D'| \leqslant 90 \cdot \alpha^{-2} \log |\mathcal{Q}|$ and an event Γ (only depending on the choice of the Laplacian variables) such that Γ has probability $1 - \beta$ and if Γ occurs, then $\mathcal{TO}(D, \alpha, b)$ has the following guarantee: whenever $\mathcal{TO}(D, \alpha, b)$ outputs l on one of the queries (q, t) in the sequence, then

1. *if $l \neq \perp$ then $l = f_t^{D'}(q) = f_t^D(q)$, and*
2. *if $l = \perp$ then $q \notin Q(t, \alpha)$.*

Proof. Let D' be the data set given by Lemma 4.2 with its "α" value set to $\alpha/3$ so that

$$\left| f^D(q) - f^{D'}(q) \right| < \alpha/3$$

for every input $q \in \mathcal{Q}$.

The event Γ is defined as the event that every Laplacian variable N_q sampled by the oracle has magnitude $|N_q| < \alpha/3$. Under the given assumption on $|D|$ in 4.5 and using basic tail bounds for the Laplacian distribution, this happens with probability $1 - \beta$.

Assuming Γ occurs, the following two statements hold:

1. Whenever the oracle outputs $l \neq \perp$ on a query (q, t), then we must have either $f^D(q) + N_q - t \geqslant 2\alpha/3$ (and thus both $f^D(q) > t + \alpha/3$ and $f^{D'}(q) > t$) or else $f^D(q) + N_q - t \leqslant -2\alpha/3$ (and thus both $f^D(q) < t - \alpha/3$ and $f^{D'}(q) < t$). This proves the first claim of the lemma.
2. Whenever $q \in Q(t, \alpha)$, then $|f^D(q) + N_q - t| \geqslant 2\alpha/3$, and therefore the oracle does not output $\perp$. This proves the second claim of the lemma.

□

4.2 Privacy-preserving reduction In this section we describe how to convert a non-private learning algorithm for threshold functions of the form f_t^D to a privacy-preserving learning algorithm for functions of the form f^D. The reduction is presented in Figure 2. We call the algorithm PrivLearn.

Input: Distribution $G \in \mathcal{GQ}$, data set D of size n, accuracy parameters $\alpha, \beta, \gamma > 0$; learning algorithm $\mathcal{L}$ for thresholds over $(\mathcal{Q}, \mathcal{GQ}, \mathcal{F})$ as in Theorem 3.1 requiring $b(n', \gamma', \beta')$ labeled examples and approximate restricted evaluation access to the target function.
Parameters: See (4.6) and (4.7).
Algorithm PrivLearn for privately learning f^D:

1. Let $\mathcal{TO}$ denote an instantiation of $\mathcal{TO}(D, \alpha/7, b_{\text{total}})$.
2. Sample b_{iter} points $\{q_j\}_{1 \leqslant j \leqslant b_{\text{iter}}}$ from G.
3. For each iteration $i \in \{1, \ldots, k\}$:
    (a) Let $t_i = i/k + 1$.
    (b) For each $q_j, j \in [b_{\text{iter}}]$ send the query (q_j, t_i) to $\mathcal{TO}$ and let l_j denote the answer. Let $B_i = \{j : l_j \neq \perp\}$.
    (c) If $\frac{|B_i|}{b_{\text{iter}}} < \frac{\gamma}{2}$, output the constant t_i function as hypothesis h and terminate the algorithm.
    (d) Run the learning algorithm $\mathcal{L}(n', t_i, \gamma', \beta')$ on the labeled examples $\{(q_j, l_j)\}_{j \in B_i}$, answering evaluation queries from $\mathcal{L}$ as follows:
        - Given a query q posed by $\mathcal{L}$, let l be the answer of $\mathcal{TO}$ on (q, t_i).
        - If $l = \perp$, then output $(0, \perp)$. Otherwise, output $(G[q] \cdot \frac{b_{\text{iter}}}{|B_i|}, l)$.
    (e) Let h_i denote the resulting hypothesis.
4. Having obtained hypotheses $h_1, \ldots, h_k$, the final hypothesis h is defined as follows: $h(q)$ equals the smallest $i \in [k]$ such that $h_i(q) = 1$ and $h_{i-1}(q) = 0$ (we take $h_0(q) = 0$ and $h_{k+1}(q) = 1$).

Figure 2: Reduction from private data release to learning thresholds (non-privately).

Setting of parameters. In the description of PrivLearn we use the following setting of parameters:
(4.6)
$$n' = \frac{4410 \cdot \log |\mathcal{Q}|}{\alpha^2}, \quad k = \left\lceil \frac{3}{\alpha} \right\rceil, \quad \gamma' = \frac{\gamma}{k}, \quad \beta' = \frac{\beta}{6k},$$

(4.7)
$$b_{\text{base}} = b(n', \gamma', \beta'), \quad b_{\text{iter}} = \frac{100 b_{\text{base}} \cdot \log(1/\beta')}{\gamma}, \quad b_{\text{total}} = 2k \cdot b_{\text{iter}}.$$

Analysis of the reduction. Throughout the analysis of the algorithm we keep all input parameters fixed so as to satisfy the assumptions of Theorem 3.1. Specifically we will need

$$|D| \geqslant \frac{210 \cdot b_{\text{total}} \cdot \log(10 b_{\text{total}}/\beta)}{\varepsilon\alpha} . \tag{4.8}$$

We have made no attempt to optimize various constants throughout.

Lemma 4.4. (Privacy) *Algorithm* PrivLearn *satisfies ε-differential privacy.*

Proof. In each iteration of the loop in Step 3 the algorithm makes at most $2b_{\text{iter}}$ queries to $\mathcal{TO}$ (there are b_{iter} calls made on the samples and at most $b_{\text{base}} \leqslant b_{\text{iter}}$ evaluation queries). But note that $\mathcal{TO}$ is instantiated with a query bound of $b_{\text{total}} = 2k b_{\text{iter}}$. Hence, it follows from Lemma 4.1 that $\mathcal{TO}$ satisfies ε-differential privacy. Since $\mathcal{TO}$ is the only way in which PrivLearn ever interacts with the data set, PrivLearn satisfies ε-differential privacy. □

We now prove that the hypothesis produced by the algorithm is indeed accurate, as formalized by the following lemma.

Lemma 4.5. (Accuracy) *With overall probability $1 - \beta$, the hypothesis h returned by* PrivLearn *satisfies*

$$\mathop{\mathbb{P}}_{q \sim G} \left\{ \left| h(q) - f^D(q) \right| \leqslant \alpha \right\} \geqslant 1 - \gamma . \tag{4.9}$$

Proof. We consider three possible cases:

1. The first case is that there exists a $t \in \{t_1, \ldots, t_k\}$ such that distribution G has at least $1 - \gamma/10$ of its mass on points that are α-close to t. In this case a Chernoff bound and the choice of $b_{\text{iter}} \gg b_{\text{base}}$ imply that with probability $1 - \beta$ the algorithm terminates prematurely and the resulting hypothesis satisfies (4.9).
2. In the second case, there exists a $t \in \{t_1, \ldots, t_k\}$ such that the probability mass G puts on points

that are α-close to t is between $1-\gamma$ and $1-\gamma/10$. In this case if the algorithm terminates prematurely then (4.9) is satisfied; below we analyze what happens assuming the algorithm does not terminate prematurely.

3. In the third case every $t \in \{t_1, \dots, t_k\}$ is such that G puts less than $1-\gamma$ of its mass on points α-close to t. In this third case if the algorithm terminates prematurely then (4.9) will not hold; however, our choice of b_{iter} implies that in this third case the algorithm terminates prematurely with probability at most $1-\beta$. As in the second case, below we will analyze what happens assuming the algorithm does not terminate prematurely.

Thus in the remainder of the argument we may assume without loss of generality that the algorithm does not terminate prematurely, i.e. it produces a full sequence of hypotheses $h_1, \dots, h_k$. Furthermore, we can assume that the distribution G places at most $1-\gamma/10$ fraction of its weight near any particular threshold t_i. This leads to the following claim, showing that in all iterations, the number of labeled examples in B_i is large enough to run the learning algorithm.

Claim 1. $\mathbb{P}\{\forall i \colon |B_i| \geqslant b_{\text{base}}\} \geqslant 1 - \frac{\beta}{3}$.

Proof. By our assumption, the probability that a sample $q \sim G$ is rejected at step t of PrivLearn is at most $\gamma/10$. By the choice of b_{iter} it follows that $|B_i| \geqslant b_{\text{base}}$ with probability $1-\beta/k$. Taking a union bound over all thresholds t completes the proof. □

The proof strategy from here on is to first analyze the algorithm on the conditional distribution that is induced by the threshold oracle. We will then pass from this conditional distribution to the actual distribution that we are interested in, namely, G.

We chose $|D|$ large enough so that we can apply Lemma 4.3 to $\mathcal{TO}$ with the "α"-setting of Lemma 4.3 set to $\alpha/7$. Let D' be the data set and Γ be the event given in the conclusion of Lemma 4.3 applied to $\mathcal{TO}$. (Note that $n' = |D'| \leqslant 7^2 \cdot 90\alpha^{-2} \log|\mathcal{Q}|$ as stated above.)

By the choice of our parameters, we have

$$\mathbb{P}\{\Gamma\} \geqslant 1 - \frac{\beta}{3}. \tag{4.10}$$

Here the probability is computed only over the internal randomness of the threshold oracle $\mathcal{TO}$ which we denote by R. Fix the randomness R of $\mathcal{TO}$ such that $R \in \Gamma$. For the sake of analysis, we can think of the randomness of the oracle as a collection of independent random variables $(N_q)_{q\in\mathcal{Q}}$ (where N_q is used to answer all queries of the form (q,t')). In particular, the behavior of the oracle would not change if we were to sample all variables $(N_q)_{q\in\mathcal{Q}}$ up front. When we fix R we thus mean that we fix N_q for all $q \in \mathcal{Q}$.

We may therefore assume for the remainder of the analysis that $\mathcal{TO}$ satisfies properties (1) and (2) of Lemma 4.3.

Let us denote by $Q_i \subseteq \mathcal{Q}$ the set of examples for which $\mathcal{TO}$ would not answer $\perp$ in Step 3 at the i-th iteration of the algorithm. Note that this is a well-defined set since we fixed the randomness of the oracle. Denote by G_i the distribution G conditioned on Q_i. Further, let $Z_i = \mathbb{P}_{q\sim G}\{q \in Q_i\}$. Observe that

$$G_i[q] = \begin{cases} G[q]/Z_i & q \in Q_i \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

The next lemma shows that PrivLearn answers evaluation queries with the desired multiplicative precision.

Lemma 4.6. *With probability $1-\beta/6k$ (over the randomness of* PrivLearn*), we have*

$$\frac{Z_i}{3} \leqslant \frac{|B_i|}{b_{\text{iter}}} \leqslant 3Z_i\,. \tag{4.12}$$

Proof. The lemma follows from a Chernoff bound with the fact that we chose $b_{\text{iter}} \gg b_{\text{base}}$. □

Assuming that (4.12) holds, we can argue that the learning algorithm in step t produces a "good" hypothesis as expressed in the next lemma.

Lemma 4.7. *Let $t \in \{t_1, \dots, t_k\}$. Conditioned on (4.12), we have that with probability $1-\beta/6k$ (over the internal randomness of the learning algorithm invoked at step i) the hypothesis h_i satisfies*

$$\mathbb{P}_{q\sim G_i}\left\{h_i(q) = f^D_{t_i}(q)\right\} \geqslant 1 - \frac{\gamma}{k}\,.$$

Proof. This follows directly from the guarantee of the learning algorithm $\mathcal{L}$ once we argue that (with the claimed probability):

1. Each example q is sampled from G_i and labeled correctly by $f^{D'}_{t_i}(q)$ and $f^{D'}_{t_i}(q) = f^D_{t_i}(q)$.

2. All evaluation queries asked by the learning algorithm are answered with the multiplicative error allowed in Definition 6.

3. The algorithm received sufficiently many, i.e., b_{base}, labeled examples.

The first claim follows from the definition of G_i, since we can sample from G_i by sampling from G and rejecting if the oracle $\mathcal{TO}$ returns $\perp$. Since Γ is assumed to hold, we can invoke property (1) of Lemma 4.3 to conclude that whenever the oracle does not return $\perp$, then its answer agrees with $f_{t_i}^{D'}(q)$ and moreover $f_{t_i}^{D'}(q) = f_{t_i}^{D}(q)$.

To see the second claim, consider an evaluation query q. We consider two cases. The first case is where the threshold oracle returns $\perp$ and PrivLearn outputs $(0, \perp)$. Note that in this case indeed G_i puts 0 weight on the query q. In the second case PrivLearn outputs $(G[q] \cdot b_{\text{iter}}/|B_i|, l)$. By (4.11) and since we assumed Γ holds, the output satisfies the desired multiplicative bound.

The third claim is a direct consequence of Claim 1. □

We conclude from the above that with probability $1-\beta/3$ (over the combined randomness of PrivLearn and of the learning algorithm), simultaneously for all $i \in [k]$ we have

$$\underset{q\sim G}{\mathbb{P}}\left\{h_i(q) \neq f_{t_i}^D(q) \;\middle|\; Q_i\right\} = \underset{q\sim G_i}{\mathbb{P}}\left\{h_i(q) \neq f_{t_i}^D(q)\right\} \leqslant \frac{\gamma}{k}\,. \tag{4.13}$$

This follows from a union bound over all k applications of Lemma 4.6 and Lemma 4.7.

We can now complete the proof of Lemma 4.5. That is, we will show that assuming (4.13) the hypothesis h satisfies

$$\underset{q\sim G}{\mathbb{P}}\left\{\left|h(q) - f^D(q)\right| \leqslant \alpha\right\} \geqslant 1 - \gamma\,.$$

Note that

1. (4.13) occurs with probability $1-\beta/3$,
2. our assumption on the threshold oracle, i.e., $R \in \Gamma$ also occurs with probability $1-\beta/3$ (over the randomness of the oracle)
3. the event in Claim 1 holds with probability $1-\beta/3$.

Hence all three events occur simultaneously with probability $1-\beta$ which is what we claimed. We proceed by assuming that all three events occurred. In the following, let

$$\mathrm{Err}_i = \{q \in \mathcal{Q} \colon h_i(q) \neq f_{t_i}^D(q)\}$$

denote the set of points on which h_i errs. We will need the following claim.

Claim 2. *Let $q \in \mathcal{Q}$. Then,*

$$\left|h(q) - f^D(q)\right| > \alpha \qquad \Longrightarrow \qquad q \in \bigcup_{i\in[k]} \mathrm{Err}_i \cap Q_i\,.$$

Proof. Arguing in the contrapositive, suppose $q \notin \bigcup_{i\in[k]} \mathrm{Err}_i \cap Q_i$. This means that for all $i \in [k]$ we have that either $q \notin \mathrm{Err}_i$ or $q \notin Q_i$.

However, we claim that there can be at most one $i \in [k]$ such that $q \notin Q_i$ meaning that q is rejected at step i. This follows from property (2) of Lemma 4.3 which asserts that if $q \notin Q_i$, then we must have $|f^D(q) - t_i| < \alpha/7$, and the fact that any two thresholds differ by at least $\alpha/3$.

Hence, under the assumption above it must be the case that $q \notin \mathrm{Err}_i$ for all but at most one $i \in [k]$. This means that all but one hypothesis h_i correctly classify q. Since the thresholds are spaced $\alpha/3$ apart, this means the hypothesis h has error at most $2\alpha/3 \leqslant \alpha$ on q. □

With the previous claim, we can finish the proof. Indeed,

$$\begin{aligned}
&\text{(using Claim 2)} \\
&\underset{q\sim G}{\mathbb{P}}\left\{\left|h(q) - f^D(q)\right| > \alpha\right\} \leqslant \underset{q\sim G}{\mathbb{P}}\left\{\bigcup_{i\in[k]} \mathrm{Err}_i \cap Q_i\right\} \\
&\text{(union bound)} \qquad \leqslant \sum_{i=1}^{k} \underset{q\sim G}{\mathbb{P}}\left\{\mathrm{Err}_i \cap Q_i\right\} \\
&\qquad = \sum_{i=1}^{k} \underset{q\sim G}{\mathbb{P}}\left\{q \in \mathrm{Err}_i \mid Q_i\right\} \underset{q\sim G}{\mathbb{P}}\left\{Q_i\right\} \\
&\qquad \leqslant \sum_{i=1}^{k} \underset{q\sim G}{\mathbb{P}}\left\{q \in \mathrm{Err}_i \mid Q_i\right\} \\
&\text{(using (4.13))} \qquad \leqslant k \cdot \frac{\gamma}{k} \\
&\qquad = \gamma\,.
\end{aligned}$$

This concludes the proof of Lemma 4.5 □

Lemma 4.4 (Privacy) together with Lemma 4.5 (Accuracy) conclude the proof of out main theorem, Theorem 3.1.

4.3 Quantitative Improvements without Membership Queries

Here we show how to shave off a factor of $1/\alpha$ in the requirement on the data set size n in Theorem 3.1. This is possible if the learning algorithm uses only sampling access to labeled examples.

Theorem 4.1. *Let $\mathcal{U}$ be a data universe, $\mathcal{Q}$ a set of query descriptions, $\mathcal{GQ}$ a set of distributions over $\mathcal{Q}$, and $P \colon \mathcal{Q} \times \mathcal{U} \to \{0,1\}$ a predicate.*

Then, there is an ε-differentially private (α, β, γ)-accurate data-release algorithm provided that there *is an algorithm $\mathcal{L}$ that (γ,β)-learns thresholds over $(\mathcal{Q}, \mathcal{GQ}', \{p_u \colon u \in \mathcal{U}\})$ using $b(n,\gamma,\beta)$ random examples; and we have*

$$n \geqslant \frac{C \cdot b(n',\gamma',\beta') \cdot \log\left(\frac{b(n',\gamma',\beta')}{\alpha\gamma\beta}\right) \cdot \log(1/\beta')}{\varepsilon\alpha\gamma},$$

where $n' = \Theta(\log|\mathcal{Q}|/\alpha^2)$, $\beta' = \Theta(\beta\alpha)$, $\gamma' = \Theta(\gamma\alpha)$ and $C > 0$ is a sufficiently large constant. If $\mathcal{L}$ runs in time $t(n,\gamma,\beta)$ then the data release algorithm runs in time $\mathrm{poly}(t(n',\gamma',\beta'), n, 1/\alpha, \log(1/\beta), 1/\gamma)$.

Proof. The proof of this theorem is identical to that of Theorem 3.1 except that we put $b_{\text{total}} = 2b_{\text{iter}}$ rather than $2kb_{\text{iter}}$. It is easy to check that the algorithm indeed makes only b_{total} *distinct queries* (in the sense of Lemma 4.1) to the threshold oracle so that privacy remains ensured. The correctness argument is identical. □

5 First Application: Data Release for Conjunctions

With Theorems 3.1 and 4.1 in hand, we can obtain new data release algorithms "automatically" from learning algorithms that satisfy the properties required by the theorem. In this section we present such data release algorithms for conjunction counting queries using learning algorithms (which require only random examples and work under any distribution) based on polynomial threshold functions.

Throughout this section we fix the query class under consideration to be monotone conjunctions, i.e. we take $\mathcal{U} = \mathcal{Q} = \{0,1\}^d$ and $P(q,u) = 1 - \bigvee_{i\colon u_i=0} q_i$.

The learning results given later in this section, together with Theorem 4.1, immediately yield:

Theorem 5.1. (Releasing conjunction queries)

1. *There is an ε-differentially private algorithm for releasing the class of monotone Boolean conjunction queries over $\mathcal{GQ} = \{$all probability distributions over $\mathcal{Q}\}$ which is (α,β,γ)-accurate and has runtime* $\mathrm{poly}(n)$ *for databases of size n provided that*

$$n \geqslant d^{O\left(d^{1/3}\log\left(\frac{d}{\alpha}\right)^{2/3}\right)} \cdot \tilde{O}\left(\frac{\log(1/\beta)^3}{\varepsilon\alpha\gamma^2}\right).$$

2. *There is an ε-differentially private algorithm for releasing the class of monotone Boolean conjunction queries over $\mathcal{GQ}_k = \{$all probability distributions over $\mathcal{Q}$ supported on $B_k = \{q \in \mathcal{Q} : q_1 + \cdots + q_d \leqslant k\}\}$ which is (α,β,γ)-accurate and has runtime* $\mathrm{poly}(n)$ *for databases of size n provided that*

$$n \geqslant d^{O\left(\sqrt{k\log\left(\frac{k\log d}{\alpha}\right)}\right)} \cdot \tilde{O}\left(\frac{\log(1/\beta)^3}{\varepsilon\alpha\gamma^2}\right).$$

These algorithms are distribution-free, and so we can apply the boosting machinery of [DRV10] to get accurate answers to *all* of the k-way conjunctions with similar database size bounds. See the discussion and Corollary 1.1 in the introduction.

In Section 5.1 we establish structural results showing that certain types of thresholded real-valued functions can be expressed as low-degree polynomial threshold functions. In Section 5.2 we state some learning results (for learning under arbitrary distributions) that follow from these representational results. Theorem 5.1 above follows immediately from combining the learning results of Section 5.2 with Theorem 4.1.

5.1 Polynomial threshold function representations

Definition 8. *Let $X \subseteq \mathcal{Q} = \{0,1\}^d$ and let f be a Boolean function $f : X \to \{0,1\}$. We say that f has a* polynomial threshold function (PTF) of degree a over X *if there is a real polynomial $A(q_1,\ldots,q_d)$ of degree a such that*

$$f(q) = sign(A(q)) \quad \textit{for all } q \in X$$

where the sign function is $sign(z) = 1$ if $z \geqslant 0$, $sign(z) = 0$ if $z < 0$.

Note that the polynomial A may be assumed without loss of generality to be multilinear since X is a subset of $\{0,1\}^d$.

5.1.1 Low-degree PTFs over sparse inputs

Let $B_k \subset \{0,1\}^d$ denote the collection of all points with Hamming weight at most k, i.e. $B_k = \{q \in \{0,1\}^d : q_1 + \cdots + q_d \leqslant k\}$. The main result of this subsection is a proof that for any $t \in [0,1]$ the function f_t^D has a low-degree polynomial threshold function over B_k.

Lemma 5.1. *Fix $t \in [0,1]$. For any database D of size n, the function f_t^D has a polynomial threshold function of degree $O\left(\sqrt{k\log n}\right)$ over the domain B_k.*

To prove Lemma 5.1 we will use the following claim:

Claim 3. *Fix $k > 0$ to be a positive integer and $\varepsilon > 0$. There is a univariate polynomial s of degree $O\left(\sqrt{k\log(1/\varepsilon)}\right)$ which is such that*

1. *$s(k) = 1$; and*
2. *$|s(j)| \leqslant \varepsilon$ for all integers $0 \leqslant j \leqslant k-1$.*

Proof. This claim was proved by Buhrman et al. [BCdWZ99], who gave a quantum algorithm which implies the existence of the claimed polynomial (see also Section 1.2 of [She09]). Here we give a self-contained construction of a polynomial s with the claimed properties that satisfies the slightly weaker degree bound $\deg(s) = O(\sqrt{k}\log(1/\varepsilon))$. We will use the univariate Chebyshev polynomial C_r of degree $r = \lceil\sqrt{k}\rceil$. Consider the polynomial

$$s(j) = \left(\frac{C_r\left(\frac{j}{k}\left(1+\frac{1}{k}\right)\right)}{C_r\left(1+\frac{1}{k}\right)}\right)^{\lceil\log(1/\varepsilon)\rceil}. \tag{5.14}$$

It is clear that if $j = k$ then $s(j) = 1$ as desired, so suppose that j is an integer $0 \leqslant j \leqslant k-1$. This implies that $(j/k)(1+1/k) < 1$. Now well-known properties of the Chebychev polynomial (see e.g. [Che66]) imply that $|C_r((j/k)(1+1/k))| \leqslant 1$ and $C_r(1+1/k) \geqslant 2$. This gives the $O(\sqrt{k}\log(1/\varepsilon))$ degree bound. □

Recall that the predicate function for a data item $u \in \{0,1\}^d$ is denoted by

$$p_u(q) = 1 - \bigvee_{i:\, u_i=0} q_i\,.$$

As an easy corollary of Claim 3 we get:

Corollary 5.1. *Fix $\varepsilon > 0$. For every $u \in \{0,1\}^d$, there is a d-variable polynomial A_u of degree $O\left(\sqrt{k}\log(1/\varepsilon)\right)$ which is such that for every $q \in B_k$,*

1. *If $p_u(q) = 1$ then $A_u(q) = 1$;*
2. *If $p_u(q) = 0$ then $|A_u(q)| \leqslant \varepsilon$.*

Proof. Consider the linear function $L(q) = k - \sum_{i:\, u_i=0} q_i$. For $q \in B_k$ we have that $L(q)$ is an integer in $\{0,\dots,k\}$, and we have $L(q) = k$ if and only if $p_u(q) = 1$. The desired polynomial is $A_u(q) = s(L(q))$. □

Proof.[Proof of Lemma 5.1] Consider the polynomial

$$A(q) = \sum_{u\in D} A_u(q)$$

where for each data item u, r_u is the polynomial from Corollary 5.1 with its "ε" parameter set to $\varepsilon = 1/(3n)$. We will show that $A(q) - (\lceil tn\rceil - 1/2)$ is the desired polynomial which gives a PTF for f_t^D over B_k.

First, consider any fixed $q \in B_k$ for which $f_t^D(q) = 1$. Such a q must satisfy $f^D(q) = j/n \geqslant t$ for some integer j, and hence $j \geqslant \lceil tn\rceil$. Corollary 5.1 now gives that $A(q) \geqslant \lceil tn\rceil - 1/3$.

Next, consider any fixed $q \in B_k$ for which $f_t^D(q) = 0$. Such a q must satisfy $f^D(q) = j/n < t$ for some integer j, and hence $j \leqslant \lceil tn\rceil - 1$. Corollary 5.1 now gives that $A(q) \leqslant \lceil tn\rceil - 2/3$. This proves the lemma. □

5.1.2 Low-degree PTFs over the entire hypercube

Taking $k = d$ in the previous subsection, the results there imply that f_t^D can be represented by a polynomial threshold function of degree $O\big(\sqrt{d\log n}\big)$ over the entire Boolean hypercube $\{0,1\}^d$. In this section we improve the degree to $O\big(d^{1/3}(\log n)^{2/3}\big)$. This result is very similar to Theorem 8 of [KOS04] (which is closely based on the main construction and result of [KS04]) but with a few differences: first, we use Claim 3 to obtain slightly improved bounds. Second, we need to use the following notion in place of the notion of the "size of a conjunction" that was used in the earlier results:

Definition 9. *The* width *of a data item $u \in D$ is defined as the number of coordinates i such that $u_i = 0$. The* width *of D is defined as the maximum width of any data item $u \in D$.*

We use the following lemma:

Lemma 5.2. *Fix any $t \in [0,1]$ and suppose that n-element database D has width w. Then f_t^D has a polynomial threshold function of degree $O\left(\sqrt{w\log n}\right)$ over the domain $\{0,1\}^d$.*

Proof. The proof follows the constructions and arguments of the previous subsection, but with "w" in place of "k" throughout (in particular the linear function $L(q)$ is now defined to be $L(q) = w - \sum_{i:\, u_i=0} q_i$). □

Lemma 5.3. *Fix any value $r \in \{1,\dots,d\}$. The function $f_t^D(q_1,\dots,q_d)$ can be expressed as a decision tree T in which*

1. *each internal node of the tree contains a variable q_i;*
2. *each leaf of T contains a function of the form $f_t^{D'}$ where $D' \subseteq D$ has width at most r;*
3. *the tree T has rank at most $(2d/r)\ln n + 1$.*

Proof.[Proof sketch.] The result follows directly from the proof of Lemma 10 in [KS04], except that we use the notion of width from Definition 9 in place of the notion of the size of a conjunction that is used in [KS04]. To see that this works, observe that since $p_u(q) = 1 - \vee_{i:\, u_i=0} q_i$, fixing $q_i = 1$ will fix all predicates p_u with $u_i = 0$ to be zero. Thus the analysis of [KS04] goes through unchanged, replacing "terms of f that have size at least r" with "data items in D that have width at least r" throughout. □

Lemma 5.4. *The function f_t^D can be represented as a polynomial threshold function of degree $O(d^{1/3}(\log n)^{2/3})$.*

Proof. The proof is nearly identical to the proof of Theorem 2 in [KS04] but with a few small changes. We take r in Lemma 5.3 to be $d^{2/3}(\log n)^{1/3}$ and now apply Lemma 5.2 to each width-r database D' at a leaf of the resulting decision tree. Arguing precisely as in Theorem 2 of [KS04] we get that f_t^D has a polynomial threshold function of degree

$$\begin{aligned}\max\left\{\tfrac{2d}{r}\ln n + 1, O\left(\sqrt{r\log n}\right)\right\} &= O\left(\sqrt{r \log n}\right)\\ &= O\left(d^{1/3}(\log n)^{2/3}\right)\end{aligned}$$

□

5.2 Learning thresholds of conjunction queries under arbitrary distributions

It is well known that using learning algorithms based on polynomial-time linear programming, having low-degree PTFs for a class of functions implies efficient PAC learning algorithms for that class under any distribution using random examples only (see e.g. [KS04, HS07]). Thus the representational results of Section 5.1 immediately give learning results for the class of threshold functions over sums of data items. We state these learning results using the terminology of our reduction below.

Theorem 5.2. *Let*

- *$\mathcal{U}$ denote the data universe $\{0,1\}^d$;*
- *$\mathcal{Q}$ denote the set of query descriptions $\{0,1\}^d$;*
- *$P(q,u) = 1 - \bigvee_{i:\, u_i=0} q_i$ denote the monotone conjunction predicate;*
- *$\mathcal{GQ}$ denote the set of all probability distributions over $\mathcal{Q}$; and*
- *$\mathcal{GQ}_k$ denote the set of all probability distributions over $\mathcal{Q}$ that are supported on $B_k = \{q \in \{0,1\}^d : q_1 + \cdots + q_d \leqslant k\}$.*

Then,

1. *1. (Learning thresholds of conjunction queries over all inputs) There is an algorithm $\mathcal{L}$ that (γ,β) learns thresholds over $(\mathcal{Q}, \mathcal{GQ}, \{p_u : u \in \mathcal{U}\})$ using $b(n,\gamma,\beta) = d^{O(d^{1/3}(\log n)^{2/3})} \cdot \tilde{O}(1/\gamma) \cdot \log(1/\beta)$ queries to an approximate distribution-restricted evaluation oracle for the target n-threshold function (in fact $\mathcal{L}$ only uses sampling access to labeled examples). The running time of $\mathcal{L}$ is* $\mathrm{poly}(b(n,\gamma,\beta))$.
2. *2. (Learning thresholds of conjunction queries over sparse inputs) There is an algorithm $\mathcal{L}$ that (γ,β) learns thresholds over $(\mathcal{Q}, \mathcal{GQ}_k, \{p_u : u \in \mathcal{U}\})$ using $b(n,\gamma,\beta) = d^{O((k\log n)^{1/2})} \cdot \tilde{O}(1/\gamma) \cdot \log(1/\beta)$ queries to an approximate distribution-restricted evaluation oracle for the target n-threshold function (in fact $\mathcal{L}$ only uses sampling access to labeled examples). The running time of $\mathcal{L}$ is* $\mathrm{poly}(b(n,\gamma,\beta))$.

Recall from the discussion at the beginning of Section 5 that these learning results, together with our reduction, give the private data release results stated at the beginning of the section.

6 Second Application: Data Release via Fourier-Based Learning

Our main result in this section is a data release algorithm for AC^0 counting queries. We obtain this algorithm via an instantiation of our reduction (Theorem 3.1), with a Fourier-based algorithm from the computational learning theory literature [JKS02]. We note that this algorithm requires the more general reduction of Theorem 3.1, rather than the simpler version of Theorem 1.1, because the underlying learning algorithm is not distribution free.

We begin by describing an algorithm for releasing parity counting queries in Section 6.1. The algorithm is base on our reduction, instantiated with Jackson's Harmonic Sieve algorithm [Jac97]. A simpler data release algorithm for parity counting queries, with better runtime and error bounds, was pointed out by an anonymous referee (see more details below). Our primary purpose in presenting this algorithm is for exposition, and as a warmup for the AC^0 counting query data release algorithm, which follows in Section 6.2.

6.1 Parity counting queries using the Harmonic Sieve [Jac97]

In this subsection we fix the query class under consideration to be the class of parity queries, i.e. we take $\mathcal{U} = \{0,1\}^d$ and $\mathcal{Q} = \{0,1\}^d$ and we take $P(q,u) = \sum_{i:u_i=1} q_i \pmod 2$ to be the parity predicate.

An anonymous referee suggested an algorithm for parity query data release over the uniform distribution. We begin by sketching this algorithm: release, using the histogram algorithm of [DMNS06], a noisy list L of items that appear frequently in the input database (say more than d times), and their (noisy) frequencies in the database. Let n' be the number of items in the database that are not in this list. For a parity $q \in \{0,1\}^d$, compute the answer with respect to L, add $n'/2$, and release the sum. For databases of size $\mathrm{poly}(d)$, this algorithm will release $\mathrm{poly}(d)$-accurate answers to all but a $\mathrm{poly}(1/d)$ fraction of the parity queries.

An alternative algorithm, which instantiates the reduction of Theorem 4.1 with Jackson's Harmonic Sieve algorithm follows. We note that the runtime and error guarantees obtained are inferior to the (significantly) simpler algorithm above. Our main purpose in presenting this algorithm is for exposition, and to serve as a warm-up for the subsequent algorithm for releasing AC^0 counting queries.

Theorem 6.1. (Releasing parity queries) *There is an ε-differentially private algorithm for releasing the class of parity queries over the uniform distribution on $\mathcal{Q}$ which is (α,β,γ)-accurate and has runtime* $\mathrm{poly}(n)$ *for databases of size n, provided that $n \geqslant \frac{\mathrm{poly}(d,1/\alpha,1/\gamma,\log(1/\beta))}{\varepsilon}$.*

This theorem is an immediate consequence of our main reduction, Theorem 3.1, and the following learning result:

Theorem 6.2. *Let*

- *$\mathcal{U}$ denote the data universe $\{0,1\}^d$;*
- *$\mathcal{Q}$ denote the set of query descriptions $\{0,1\}^d$;*
- *$P(q,u) = \sum_{i:u_i=1} q_i$ (mod 2) denote the parity predicate; and*
- *$\mathcal{GQ}$ contains only the uniform distribution over $\mathcal{Q}$.*

Then there is an algorithm $\mathcal{L}$ that (γ,β) learns thresholds over $(\mathcal{Q}, \mathcal{GQ}', \{p_u : u \in \mathcal{U}\})$ where $\mathcal{GQ}'$ is the $(2/\gamma)$-smooth extension of $\mathcal{GQ}$. Algorithm $\mathcal{L}$ uses $b(n,\gamma,\beta) =$ $\mathrm{poly}(d,n,1/\gamma)\cdot\log(1/\beta)$ *queries to an approximate G-restricted evaluation oracle for the target n-threshold function when it is learning with respect to a distribution $G \in \mathcal{GQ}'$. The running time of $\mathcal{L}$ is* $\mathrm{poly}(b(n,\gamma,\beta))$.

Proof. The claimed algorithm $\mathcal{L}$ is essentially Jackson's Harmonic Sieve algorithm [Jac97] for learning Majority of Parities; however, a bit of additional analysis of the algorithm is needed as we now explain.

When Jackson's results on the Harmonic Sieve are expressed in our terminology, they give Theorem 6.2 exactly as stated above except for one issue which we now describe. Let G' be any distribution in the $(2/\gamma)$-smooth extension $\mathcal{GQ}'$ of the uniform distribution. In Jackson's analysis, when it is learning a target function f under distribution G', the Harmonic Sieve is given black-box oracle access to f, sampling access to the distribution G', and access to a c-approximation to an evaluation oracle for G', in the following sense: there is some fixed constant $c \in [1/3, 3]$ such that when the oracle is queried on $q \in \mathcal{Q}$, it outputs $c \cdot G'[q]$. This is a formally more powerful type of access to the underlying distribution G' than is allowed in Theorem 6.2 since Theorem 6.2 only gives $\mathcal{L}$ access to an approximate G'-restricted evaluation oracle for the target function (recall Definition 6). To be more precise, the only difference is that with the Sieve's black-box oracle access to the target function f it is a priori possible for a learning algorithm to query f even on points where the distribution G' puts zero probability mass, whereas such queries are not allowed for $\mathcal{L}$. Thus to prove Theorem 6.2 it suffices to argue that the Harmonic Sieve algorithm, when it is run under distribution G', never needs to make queries on points $q \in \mathcal{Q}$ that have $G'[q] = 0$.

Fortunately, this is an easy consequence of the way the Harmonic Sieve algorithm works. Instead of actually using black-box oracle queries for f, the algorithm actually only ever makes oracle queries to the function $g(q) = 2^d \cdot f(q) \cdot D'[q]$, where D' is a c-approximation to an evaluation oracle for a distribution G'' which is a smooth extension of G'. (See the discussion in Sections 4.1 and 4.2 of [Jac97], in particular Steps 16-18 of the `HS` algorithm of Figure 4 and Steps 3 and 5 of the `WDNF` algorithm of Figure 3.) By the definition of a smooth extension, if q is such that $G'[q] = 0$ then $G''[q]$ also equals 0, and consequently $g(q) = 0$ as well. Thus it is straightforward to run the Harmonic Sieve using access to an approximate G'-restricted evaluation oracle: if $G'[q]$ returns 0 then "0" is the correct value of $g(q)$, and otherwise the oracle provides precisely the information that would be available for the Sieve in Jackson's original formulation. □

6.2 AC^0 queries using [JKS02] Fix $\mathcal{U} = \{0,1\}^d$ and $\mathcal{Q} = \{0,1\}^d$. In this subsection we show that our reduction enables us to do efficient private data release for quite a broad class of queries, namely any query computed by a constant-depth circuit.

In more detail, let $P(q,u)\colon \{0,1\}^d \times \{0,1\}^d \to \{0,1\}$ be any predicate that is computed by a circuit of depth $\ell = O(1)$ and size $\mathrm{poly}(d)$. Our data release result for such queries is the following:

Theorem 6.3. (Releasing AC^0 queries) *Let $\mathcal{GQ}$ be the set containing the uniform distribution and let $\mathcal{U}, \mathcal{Q}, P$ be as described above. There is an ε-differentially $(\mathcal{U}, \mathcal{Q}, \mathcal{GQ}, P)$ data release algorithm that is (α, β, γ)-accurate and has runtime $\mathrm{poly}(n)$ for databases of size n, provided that*

$$n \geqslant d^{O\left(\log^\ell\left(\frac{d}{\alpha\gamma}\right)\right)} \cdot \tilde{O}\left(\frac{\log(1/\beta)^3}{\varepsilon\alpha^2\gamma}\right).$$

See the introduction for a discussion of this result. We observe that given any fixed P as described above, for any given $u \in \mathcal{U} = \{0,1\}^d$ the function $p_u(q)$ is computed by a circuit of depth ℓ and size $\mathrm{poly}(d)$ over the input bits $q_1, \ldots, q_d$. Hence Theorem 6.3 is an immediate consequence of Theorem 3.1 and the following learning result, which describes the performance guarantee of the quasipolynomial-time algorithm of Jackson et al. [JKS02] for learning Majority-of-Parity in our language:

Theorem 6.4. (Theorem 9 of [JKS02]) *Let*

- *$\mathcal{U}$ denote the data universe $\{0,1\}^d$;*
- *$\mathcal{Q}$ denote the set of query descriptions $\{0,1\}^d$;*
- *$P(q,u)$ be any fixed predicate computed by an AND/OR/NOT circuit of depth $\ell = O(1)$ and size $\mathrm{poly}(d)$;*
- *$\mathcal{GQ}$ contains only the uniform distribution over $\mathcal{Q}$; and*
- *$\mathcal{F}$ be the set of all AND/OR/NOT circuits of depth ℓ and size $\mathrm{poly}(d)$.*

Then there is an algorithm $\mathcal{L}$ that (γ, β) learns n-thresholds over $(\mathcal{Q}, \mathcal{GQ}', \mathcal{F})$ where $\mathcal{GQ}'$ is the $(2/\gamma)$-smooth extension of $\mathcal{GQ}$. Algorithm $\mathcal{L}$ uses approximate distribution restricted oracle access to the function, uses $b(n,\gamma,\beta) = d^{O(\log^\ell(nd/\gamma))} \cdot \log(1/\beta)$ samples and calls to the evaluation oracle, and runs in time $t(n,\gamma,\beta) = d^{O(\log^\ell(nd/\gamma))} \cdot \log(1/\beta)$.

We note that Theorem 9 of [JKS02], as stated in that paper, only deals with learning majority-of-AC^0 circuits under the uniform distribution: it says that an n-way Majority of depth-ℓ, size-$\mathrm{poly}(d)$ circuits over $\{0,1\}^d$ can be learned to accuracy γ and confidence β under the uniform distribution, using random examples only, in time $d^{O(\log^\ell(nd/\gamma))} \cdot \log(1/\beta)$. However, the boosting-based algorithm of [JKS02] is identical in its high-level structure to Jackson's Harmonic Sieve; the only difference is that the [JKS02] weak learner simply performs an exhaustive search over all low-weight parity functions to find a weak hypothesis that has non-negligible correlation with the target, whereas the Harmonic Sieve uses a more sophisticated membership-query algorithm (that is an extension of the algorithm of Kushilevitz and Mansour [KM93]). Arguments identical to the ones Jackson gives for the Harmonic Sieve (in Section 7.1 of [Jac97]) can be applied unchanged to the [JKS02] algorithm, to show that it extends, just like the Harmonic Sieve, to learning under smooth distributions if it is provided with an approximate evaluation oracle for the smooth distribution. In more detail, these arguments show that for any C-smooth distribution G', given sampling access to labeled examples by (G', f) (where f is the target n-way Majority of depth-ℓ, size-$\mathrm{poly}(d)$ circuits) and approximate evaluation access to G', the [JKS02] algorithm learns f to accuracy γ and confidence β under G' in time $d^{O(\log^\ell(Cnd/\gamma))} \cdot \log(1/\beta)$ This is the result that is restated in our data privacy language above (note that the smoothness parameter there is $C = 2/\gamma$).

7 Conclusion and open problems

This work put forward a new reduction from privacy-preserving data analysis to learning thresholds. Instantiating this reduction with various different learning algorithms, we obtained new data release algorithms for a variety of query classes. One notable improvement was for the database size (or error) in distribution-free release of conjunctions and k-way conjunctions. Given these new results, we see no known obstacles for even more dramatic improvements on this central question. In particular, we conclude with the following open question.

Open Question 1. *Is there a differentially private distribution-free data release algorithm (with constant error, e.g., $\alpha = 1/100$) for conjunctions or k-way conjunctions that works for databases of size $\mathrm{poly}(d)$ and runs in time $\mathrm{poly}(n)$ (or $\mathrm{poly}(n, d^k)$ for the case of k-way conjunctions)?*

Note that such an algorithm for k-way conjunctions would also imply, via boosting [DRV10], that we can privately release *all* k-way conjunctions in time $\mathrm{poly}(n, d^k)$, provided that $|D| \geqslant \mathrm{poly}(d)$.

8 Acknowledgements

We thank the anonymous SODA 2012 referees for their insightful comments. We especially thank an anonymous referee for pointing out the alternative simpler algorithm for releasing parity counting queries, outlined in Section 6.1.

References

[Ajt83] Miklós Ajtai. $\Sigma^1[1]$-formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–43, 1983.

[BCD+07] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proc. 26th Symposium on Principles of Database Systems (PODS)*, pages 273–282. ACM, 2007.

[BCdWZ99] Harry Buhrman, Richard Cleve, Ronald de Wolf, and Christof Zalka. Bounds for small-error and zero-error quantum algorithms. In *Proc. 40th Foundations of Computer Science (FOCS)*, pages 358–368. IEEE, 1999.

[BDMN05] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *Proc. 24th Symposium on Principles of Database Systems (PODS)*, pages 128–138. ACM, 2005.

[BLR08] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proc. 40th STOC*, pages 609–618. ACM, 2008.

[Che66] Elliott W. Cheney. *Introduction to Approximation Theory*. McGraw-Hill, New York, New York, 1966.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. 3rd TCC*, pages 265–284. Springer, 2006.

[DNR+09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proc. 41st STOC*, pages 381–390. ACM, 2009.

[DRV10] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Proc. 51st Foundations of Computer Science (FOCS)*. IEEE, 2010.

[GHRU11] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jon Ullman. Privately releasing conjunctions and the statistical query barrier. In *Proc. 43nd STOC*, pages 803–812. ACM, 2011.

[HR10] Moritz Hardt and Guy Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Proc. 51st Foundations of Computer Science (FOCS)*, pages 61–70. IEEE, 2010.

[HS07] Lisa Hellerstein and Rocco A. Servedio. On PAC learning algorithms for rich boolean function classes. *Theoretical Computer Science*, 384(1):66–76, 2007.

[Jac97] Jeffrey C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.

[JKS02] Jeffrey Jackson, Adam Klivans, and Rocco A. Servedio. Learnability beyond AC^0. In *Proc. 34th STOC*, pages 776–784. ACM, 2002.

[KLN+08] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *Proc. 49th Foundations of Computer Science (FOCS)*, pages 531–540. IEEE, 2008.

[KM93] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. on Computing*, 22(6):1331–1348, 1993.

[KMR+94] Michael J. Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions. In *Proc. 26th STOC*, pages 273–282. ACM, 1994.

[KOS04] Adam Klivans, Ryan O'Donnell, and Rocco A. Servedio. Learning intersections and thresholds of halfspaces. *Journal of Computer & System Sciences*, 68(4):808–840, 2004.

[KRSU10] Shiva Kasiviswanathan, Mark Rudelson, Adam Smith, and Jonathan Ullman. The price of privately releasing contingency tables and the spectra of random matrices with correlated rows. In *Proc. 42nd STOC*, pages 775–784. ACM, 2010.

[KS04] Adam Klivans and Rocco A. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *Journal of Computer & System Sciences*, 68(2):303–318, 2004.

[Nao96] Moni Naor. Evaluation may be easier than generation. In *Proc. 28th STOC*, pages 74–83. ACM, 1996.

[RR10] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proc. 42nd STOC*, pages 765–774. ACM, 2010.

[She09] Alexander A. Sherstov. The intersection of two halfspaces has high threshold degree. In *Proc. 50th Foundations of Computer Science (FOCS)*. IEEE, 2009.

[UV11] Jonathan Ullman and Salil P. Vadhan. Pcps and the hardness of generating private synthetic data. In *TCC*, pages 400–416. Springer, 2011.

[Val84] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.