Faster Algorithms for Privately Releasing Marginals

Faster Algorithms for Privately Releasing Marginals*

Justin Thaler**, Jonathan Ullman***, and Salil Vadhan†

School of Engineering and Applied Sciences &
Center for Research on Computation and Society
Harvard University, Cambridge, MA
{jthaler,jullman,salil}@seas.harvard.edu

Abstract. We study the problem of releasing *k-way marginals* of a database $D \in (\{0,1\}^d)^n$, while preserving differential privacy. The answer to a k-way marginal query is the fraction of D's records $x \in \{0,1\}^d$ with a given value in each of a given set of up to k columns. Marginal queries enable a rich class of statistical analyses of a dataset, and designing efficient algorithms for privately releasing marginal queries has been identified as an important open problem in private data analysis (cf. Barak et. al., PODS '07).

We give an algorithm that runs in time $d^{O(\sqrt{k})}$ and releases a private summary capable of answering any k-way marginal query with at most $\pm.01$ error on every query as long as $n \geq d^{O(\sqrt{k})}$. To our knowledge, ours is the first algorithm capable of privately releasing marginal queries with non-trivial worst-case accuracy guarantees in time substantially smaller than the number of k-way marginal queries, which is $d^{\Theta(k)}$ (for $k \ll d$).

1 Introduction

Consider a database $D \in (\{0,1\}^d)^n$ in which each of the $n = |D|$ rows corresponds to an individual's record, and each record consists of d binary attributes. The goal of privacy-preserving data analysis is to enable rich statistical analyses on the database while protecting the privacy of the individuals. In this work, we seek to achieve *differential privacy* [6], which guarantees that no individual's data has a significant influence on the information released about the database.

One of the most important classes of statistics on a dataset is its *marginals*. A marginal query is specified by a set $S \subseteq [d]$ and a pattern $t \in \{0,1\}^{|S|}$. The query asks, "What fraction of the individual records in D has each of the attributes

* A full version of this paper appears on arXiv [19].

** http://seas.harvard.edu/~jthaler. Supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program, and in part by NSF grants CCF-0915922 and IIS-0964473.

*** http://seas.harvard.edu/~jullman. Supported by NSF grant CNS-0831289 and a gift from Google, Inc.

† http://seas.harvard.edu/~salil. Supported by NSF grant CNS-0831289 and a gift from Google, Inc.

$j \in S$ set to t_j?" A major open problem in privacy-preserving data analysis is to *efficiently* create a differentially private summary of the database that enables analysts to answer each of the 3^d marginal queries. A natural subclass of marginals are *k-way marginals*, the subset of marginals specified by sets $S \subseteq [d]$ such that $|S| \leq k$.

Privately answering marginal queries is a special case of the more general problem of privately answering *counting queries* on the database, which are queries of the form, "What fraction of individual records in D satisfy some property q?" Early work in differential privacy [5,2,6] showed how to approximately answer any set of of counting queries $\mathcal{Q}$ by perturbing the answers with appropriately calibrated noise, providing good accuracy (say, within $\pm.01$ of the true answer) as long as $|D| \gtrsim |\mathcal{Q}|^{1/2}$.

In a setting where the queries arrive online, or are known in advance, it may be reasonable to assume that $|D| \gtrsim |\mathcal{Q}|^{1/2}$. However, many situations necessitate a non-interactive data release, where the data owner computes and publishes a single differentially private summary of the database that enables analysts to answer a large class of queries, say all k-way marginals for a suitable choice of k. In this case $|\mathcal{Q}| = d^{\Theta(k)}$, and it may be impractical to collect enough data to ensure $|D| \gtrsim |\mathcal{Q}|^{1/2}$. Fortunately, the remarkable work of Blum et. al. [3] and subsequent refinements [7,9,17,13,12,11], have shown how to privately release approximate answers to any set of counting queries, even when $|\mathcal{Q}|$ is *exponentially larger* than $|D|$. For example, these algorithms can release all k-way marginals as long as $|D| \geq \tilde{\Theta}(k\sqrt{d})$. Unfortunately, all of these algorithms have running time at least 2^d, even when $|\mathcal{Q}|$ is the set of 2-way marginals (and this is inherent for algorithms that produce "synthetic data" [20]; as discussed below).

Given this state of affairs, it is natural to seek *efficient* algorithms capable of privately releasing approximate answers to marginal queries even when $|D| \ll d^k$. A recent series of works [10,4,14] have shown how to privately release answers to k-way marginal queries with small *average error* (over various distributions on the queries) with both running time and minimum database size much smaller than d^k (e.g. $d^{O(1)}$ for product distributions [10,4] and $\min\{d^{O(\sqrt{k})}, d^{O(d^{1/3})}\}$ for arbitrary distributions [14]). Hardt et. al. [14] also gave an algorithm for privately releasing k-way marginal queries with small *worst-case error* and minimum database size much smaller than d^k. However the running time of their algorithm is still $d^{\Theta(k)}$, which is polynomial in the number of queries.

In this paper, we give the first algorithms capable of releasing k-way marginals up to small worst-case error, with both running time and minimum database size substantially smaller than d^k. Specifically, we show how to create a private summary in time $d^{O(\sqrt{k})}$ that gives approximate answers to all k-way marginals as long as $|D|$ is at least $d^{O(\sqrt{k})}$. When $k = d$, our algorithm runs in time $2^{\tilde{O}(\sqrt{d})}$, and is the first algorithm for releasing *all* marginals in time $2^{o(d)}$.

1.1 Our Results and Techniques

In this paper, we give faster algorithms for releasing marginals and other classes of counting queries.

Theorem 1 (Releasing Marginals). *There exists a constant C such that for every $k, d, n \in \mathbb{N}$ with $k \leq d$, every $\alpha \in (0,1]$, and every $\varepsilon > 0$, there is an ε-differentially private sanitizer that, on input a database $D \in (\{0,1\}^d)^n$, runs in time $|D| \cdot d^{C\sqrt{k}\log(1/\alpha)}$ and releases a summary that enables computing each of the k-way marginal queries on D up to an additive error of at most α, provided that $|D| \geq d^{C\sqrt{k}\log(1/\alpha)}/\varepsilon$.*

For notational convenience, we focus on *monotone k-way disjunction queries*. However, our results extend straightforwardly to general non-monotone k-way disjunction queries (see Section 4.1), which are equivalent to k-way marginals. A monotone k-way disjunction is specified by a set $S \subseteq [d]$ of size k and asks what fraction of records in D have at least one of the attributes in S set to 1.

Our algorithm is inspired by a series of works reducing the problem of private query release to various problems in learning theory. One ingredient in this line of work is a shift in perspective introduced by Gupta, Hardt, Roth, and Ullman [10]. Instead of viewing disjunction queries as a set of functions on the database, they view the database as a function $f_D \colon \{0,1\}^d \to [0,1]$, in which each vector $s \in \{0,1\}^d$ is interpreted as the indicator vector of a set $S \subseteq [d]$, and $f_D(s)$ equals the evaluation of the disjunction specified by S on the database D. They use the structure of the functions f_D to privately learn an approximation g_D that has small *average error* over any product distribution on disjunctions.[1]

Cheraghchi, Klivans, Kothari, and Lee [4] observed that the functions f_D can be approximated by a low-degree polynomial with small average error over the uniform distribution on disjunctions. They then use a private learning algorithm for low-degree polynomials to release an approximation to f_D; and thereby obtain an improved dependence on the accuracy parameter, as compared to [10].

Hardt, Rothblum, and Servedio [14] observe that f_D is itself an average of disjunctions (each row of D specifies a disjunction of bits in the indicator vector $s \in \{0,1\}^d$ of the query), and thus develop private learning algorithms for threshold of sums of disjunctions. These learning algorithms are also based on low-degree approximations of sums of disjunctions. They show how to use their private learning algorithms to obtain a sanitizer with small average error over *arbitrary distributions* with running time and minimum database size $d^{O(\sqrt{k})}$. They then are able to apply the private boosting technique of Dwork, Rothblum, and Vadhan [9] to obtain worst-case accuracy guarantees. Unfortunately, the boosting step incurs a blowup of d^k in the running time.

We improve the above results by showing how to *directly* compute (a noisy version of) a polynomial p_D that is privacy-preserving and still approximates f_D on *all* k-way disjunctions, as long as $|D|$ is sufficiently large. Specifically, the running time and the database size requirement of our algorithm are both polynomial in the number of monomials in p_D, which is $d^{O(\sqrt{k})}$. By "directly", we mean that we compute p_D from the database D itself and perturb its coefficients,

[1] In their learning algorithm, privacy is defined with respect to the rows of the database D that defines f_D, not with respect to the examples given to the learning algorithm (unlike earlier works on "private learning" [15]).

rather than using a learning algorithm. Our construction of the polynomial p_D uses the same low-degree approximations exploited by Hardt et. al. in the development of their private learning algorithms.

In summary, the main difference between prior work and ours is that prior work used learning algorithms that have restricted access to the database, and released the hypothesis output by the learning algorithm. In contrast, we do not make use of any learning algorithms, and give our release algorithm direct access to the database. This enables our algorithm to achieve a worst-case error guarantee while maintaining a minimal database size and running time much smaller than the size of the query set. Our algorithm is also substantially simpler than that of Hardt et. al.

We also consider other families of counting queries. We define the class of *r-of-k queries*. Like a monotone k-way disjunction, an r-of-k query is defined by a set $S \subseteq [d]$ such that $|S| \leq k$. The query asks what fraction of the rows of D have at least r of the attributes in S set to 1. For $r = 1$, these queries are exactly monotone k-way disjunctions, and r-of-k queries are a strict generalization.

Theorem 2 (Releasing r-of-k Queries). *For every $r, k, d, n \in \mathbb{N}$ with $r \leq k \leq d$, every $\alpha \in (0, 1]$, and every $\varepsilon > 0$ there is an ε-differentially private sanitizer that, on input a database $D \in (\{0,1\}^d)^n$, runs in time $|D| \cdot d^{\tilde{O}\left(\sqrt{rk \log(1/\alpha)}\right)}$ and releases a summary that enables computing each of the r-of-k queries on D up to an additive error of at most α, provided that $|D| \geq d^{\tilde{O}\left(\sqrt{rk \log(1/\alpha)}\right)}/\varepsilon$.*

Note that monotone k-way disjunctions are just r-of-k queries where $r = 1$, thus Theorem 2 implies a release algorithm for disjunctions with quadratically better dependence on $\log(1/\alpha)$, at the cost of slightly worse dependence on k (implicit in the switch from $O(\cdot)$ to $\tilde{O}(\cdot)$).

Finally, we present a sanitizer for privately releasing databases in which the *rows* of the database are interpreted as decision lists, and the *queries* are inputs to the decision lists. That is, instead of each record in D being a string of d attributes, each record is an element of the set $\mathrm{DL}_{k,m}$, which consists of all length-k decision lists over m input variables. (See Section 4.3 for a precise definition.) A query is specified by a string $y \in \{0,1\}^d$ and asks "What fraction of database participants would make a certain decision based on the input y?"

As an example application, consider a database that allows high school students to express their preferences for colleges in the form of a decision list. For example, a student may say, "If the school is ranked in the top ten nationwide, I am willing to apply to it. Otherwise, if the school is rural, I am unwilling to apply. Otherwise, if the school has a good basketball team then I am willing to apply to it." And so on. Each student is allowed to use up to k attributes out of a set of m binary attributes. Our sanitizer allows any college (represented by its m binary attributes) to determine the fraction of students willing to apply.

Theorem 3 (Releasing Decision Lists). *For any $k, m \in \mathbb{N}$ s.t. $k \leq m$, any $\alpha \in (0, 1]$, and any $\varepsilon > 1/n$, there is an ε-differentially private sanitizer with running time $m^{\tilde{O}\left(\sqrt{k} \log(1/\alpha)\right)}$ that, on input a database $D \in (\mathrm{DL}_{k,m})^n$, releases*

a summary that enables computing any length-k decision list query up to an additive error of at most α on every query, provided that $|D| \geq m^{\tilde{O}\left(\sqrt{k}\log(1/\alpha)\right)}/\varepsilon$.

For comparison, we note that all the results on releasing k-way disjunctions (including ours) also apply to a dual setting where the database *records* specify a k-way disjunction over m bits and the *queries* are m-bit strings (in this setting m plays the role of d). Theorem 3 generalizes this dual version of Theorem 1, as length-k decision lists are a strict generalization of k-way disjunctions.

We prove the latter two results (Theorems 2 and 3) using the same approach outlined for marginals (Theorem 1), but with different low-degree polynomial approximations appropriate for the different types of queries.

Paper	Running Time	Database Size	Error Type[a]	Synthetic Data?
[5,8,2,6]	$d^{O(k)}$	$O(d^{k/2}/\alpha)$	Worst case	N
[1]	$2^{O(d)}$	$O(d^{k/2}/\alpha)$	Worst case	Y
[3,7,9,12]	$2^{O(d)}$	$\tilde{O}(k\sqrt{d}/\alpha^2)$	Worst case	Y
[10]	$d^{\tilde{O}(1/\alpha^2)}$	$d^{\tilde{O}(1/\alpha^2)}$	Product Dists.	N
[4]	$d^{O(\log(1/\alpha))}$	$d^{O(\log(1/\alpha))}$	Uniform Dist.[b]	N
[14]	$d^{O(d^{1/3}\log(1/\alpha))}$	$d^{O(d^{1/3}\log(1/\alpha))}$	Any Dist.	N
[14]	$d^{O(k)}$	$d^{O(d^{1/3}\log(1/\alpha))}$	Worst case	N
[14]	$d^{O(\sqrt{k}\log(1/\alpha))}$	$d^{O(\sqrt{k}\log(1/\alpha))}$	Any Dist.	N
[14]	$d^{O(k)}$	$d^{O(\sqrt{k}\log(1/\alpha))}$	Worst case	N
This paper	$d^{O(\sqrt{k}\log(1/\alpha))}$	$d^{O(\sqrt{k}\log(1/\alpha))}$	Worst case	N

Table 1. Summary of prior results on differentially private release of k-way marginals. The database size column indicates the minimum database size required to release answers to k-way marginals up to an additive error of α. For clarity, we ignore the dependence on the privacy parameters and the failure probability of the algorithms. Notice that this paper contains the first algorithm capable of releasing k-way marginals with running time and worst-case error substantially smaller than the number of queries.

[a] *Worst case* error indicates that the accuracy guarantee holds for every marginal. The other types of error indicate that accuracy holds for random marginals over a given distribution from a particular class of distributions (e.g. product distributions).

[b] The results of [4] apply only to the uniform distribution over *all* marginals.

On Synthetic Data. An attractive type of summary is a *synthetic database*. A synthetic database is a new database $\widehat{D} \in (\{0,1\}^d)^{\widehat{n}}$ whose rows are "fake", but such that $\widehat{D}$ approximately preserves many of the statistical properties of the database D (e.g. all the marginals). Some of the previous work on counting query release has provided synthetic data, starting with Barak et. al. [1] and including [3,7,9,12].

Unfortunately, Ullman and Vadhan [20] (building on [7]) have shown that no differentially private sanitizer with running time $d^{O(1)}$ can take a database $D \in (\{0,1\}^d)^n$ and output a private synthetic database $\widehat{D}$, all of whose 2-way marginals are approximately equal to those of D (assuming the existence of one-way functions). They also showed that there is a constant $k \in \mathbb{N}$ such that no

differentially private sanitizer with running time $2^{d^{1-\Omega(1)}}$ can output a private synthetic database, all of whose k-way marginals are approximately equal to those of D (under stronger cryptographic assumptions).

When $k = d$, our sanitizer runs in time $2^{\tilde{O}(\sqrt{d})}$ and releases a private summary that enables an analyst to approximately answer any marginal query on D. Prior to our work it was not known how to release *any* summary enabling approximate answers to all marginals in time $2^{d^{1-\Omega(1)}}$. Thus, our results show that releasing a private summary for all marginal queries can be done considerably more efficiently if we do not require the summary to be a synthetic database (under the hardness assumptions made in [20]).

2 Preliminaries

2.1 Differentially Private Sanitizers

Let a *database* $D \in \mathcal{X}^n$ be a collection of n rows $x^{(1)}, \ldots, x^{(n)}$ from a *data universe* $\mathcal{X}$. We say that two databases $D_1, D_2 \in \mathcal{X}^n$ are *adjacent* if they differ only on a single row, and we denote this by $D_1 \sim D_2$.

A *sanitizer* $\mathcal{A} : \mathcal{X}^n \to \mathcal{R}$ takes a database as input and outputs some data structure in $\mathcal{R}$. We are interested in sanitizers that satisfy *differential privacy.*

Definition 4 (Differential Privacy [6]). *A sanitizer* $\mathcal{A}\colon \mathcal{X}^n \to \mathcal{R}$ *is* (ε, δ)-differentially private *if for every two adjacent databases* $D, D' \in \mathcal{X}^n$ *and every subset* $S \subseteq \mathcal{R}$, $\Pr[\mathcal{A}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{A}(D') \in S] + \delta$. *In the case where* $\delta = 0$ *we say that* $\mathcal{A}$ *is* ε-differentially private.

Since a sanitizer that always outputs $\perp$ satisfies Definition 4, we also need to define what it means for a sanitizer to be accurate. In particular, we are interested in sanitizers that give accurate answers to *counting queries.* A counting query is defined by a boolean predicate $q\colon \mathcal{X} \to \{0, 1\}$. We define the evaluation of the query q on a database $D \in \mathcal{X}^n$ to be $q(D) = \frac{1}{n}\sum_{i=1}^{n} q(x^{(i)})$. We use $\mathcal{Q}$ to denote a set of counting queries.

Since $\mathcal{A}$ may output an arbitrary data structure, we must specify how to answer queries in $\mathcal{Q}$ from the output $\mathcal{A}(D)$. Hence, we require that there is an *evaluator* $\mathcal{E}\colon \mathcal{R} \times \mathcal{Q} \to \mathbb{R}$ that estimates $q(D)$ from the output of $\mathcal{A}(D)$. For example, if $\mathcal{A}$ outputs a vector of "noisy answers" $Z = (q(D) + Z_q)_{q \in \mathcal{Q}}$, where Z_q is a random variable for each $q \in \mathcal{Q}$, then $\mathcal{R} = \mathbb{R}^{\mathcal{Q}}$ and $\mathcal{E}(Z, q)$ is the q-th component of Z. Abusing notation, we write $q(Z)$ and $q(\mathcal{A}(D))$ as shorthand for $\mathcal{E}(Z, q)$ and $\mathcal{E}(\mathcal{A}(D), q)$, respectively. Since we are interested in the efficiency of the sanitization process as a whole, when we refer to the running time of $\mathcal{A}$, we also include the running time of the evaluator $\mathcal{E}$. We say that $\mathcal{A}$ is "accurate" for the query set $\mathcal{Q}$ if the values $q(\mathcal{A}(D))$ are close to the answers $q(D)$. Formally,

Definition 5 (Accuracy). *An output* Z *of a sanitizer* $\mathcal{A}(D)$ *is* α-accurate *for the query set* $\mathcal{Q}$ *if* $|q(Z) - q(D)| \leq \alpha$ *for every* $q \in \mathcal{Q}$. *A sanitizer is* (α, β)-accurate *for the query set* $\mathcal{Q}$ *if for every database* D,

$$\Pr[\forall q \in \mathcal{Q}, |q(\mathcal{A}(D)) - q(D)| \leq \alpha] \geq 1 - \beta,$$

where the probability is taken over the coins of $\mathcal{A}$.

We will make use of the *Laplace mechanism*. Let $\mathrm{Lap}^k(\sigma)$ denote a draw from the random variable over $\mathbb{R}^k$ in which each coordinate is chosen independently according to the density function $\mathrm{Lap}_\sigma(x) \propto e^{-|x|/\sigma}$. Let $D \in \mathcal{X}^n$ be a database and $g : \mathcal{X}^n \to \mathbb{R}^k$ be a function such that for every pair of adjacent databases $D \sim D'$, $\|g(D) - g(D')\|_\infty \leq \Delta$. Then we have the following two theorems:

Lemma 6 (Laplace Mechanism, ε-Differential Privacy [6]). *For D, g, k, Δ as above, the mechanism $\mathcal{A}(D) = g(D) + \mathrm{Lap}^k(\Delta k/\varepsilon)$ satisfies ε-differential privacy. Furthermore, for any $\beta > 0$, $\Pr_{\mathcal{A}}\left[\|g(D) - \mathcal{A}(D)\|_1 \leq \alpha\right] \geq 1 - \beta$, for $\alpha = 2\Delta k^2 \log(k/\beta)/\varepsilon$.*

The choice of the L_1 norm in the accuracy guarantee of the lemma is for convenience, and doesn't matter for the parameters of Theorems 1-3 (except for the hidden constants).

2.2 Query Function Families

We take the approach of Gupta et. al. [10] and think of the database D as specifying a function f_D mapping queries q to their answers $q(D)$, which we call the *$\mathcal{Q}$-representation of D*. We now describe this transformation more formally:

Definition 7 ($\mathcal{Q}$-Function Family). *Let $\mathcal{Q} = \{q_y\}_{y \in Y_\mathcal{Q} \subseteq \{0,1\}^m}$ be a set of counting queries on a data universe $\mathcal{X}$, where each query is indexed by an m-bit string. We define the* index set of $\mathcal{Q}$ *to be the set $Y_\mathcal{Q} = \{y \in \{0,1\}^m \mid q_y \in \mathcal{Q}\}$.*

*We define the $\mathcal{Q}$-*function family *$\mathcal{F}_\mathcal{Q} = \{f_x : \{0,1\}^m \to \{0,1\}\}_{x \in \mathcal{X}}$ as follows: For every possible database row $x \in \mathcal{X}$, the function $f_{\mathcal{Q},x} : \{0,1\}^m \to \{0,1\}$ is defined as $f_{\mathcal{Q},x}(y) = q_y(x)$. Given a database $D \in \mathcal{X}^n$ we define the function $f_{\mathcal{Q},D} : \{0,1\}^m \to [0,1]$ where $f_{\mathcal{Q},D}(q) = \frac{1}{n}\sum_{i=1}^n f_{\mathcal{Q},x^{(i)}}(q)$. When $\mathcal{Q}$ is clear from context we will drop the subscript $\mathcal{Q}$ and simply write f_x, f_D, and $\mathcal{F}$.*

For some intuition about this transformation, when the queries are monotone k-way disjunctions on a database $D \in (\{0,1\}^d)^n$, the queries are defined by sets $S \subseteq [d]$, $|S| \leq k$. In this case each query can be represented by the d-bit indicator vector of the set S, with at most k non-zero entries. Thus we can take $m = d$ and $Y_\mathcal{Q} = \left\{y \in \{0,1\}^d \mid \sum_{j=1}^d y_j \leq k\right\}$.

2.3 Polynomial Approximations

An m-variate real polynomial $p \in \mathbb{R}[y_1, \ldots, y_m]$ of *degree* t and *(L_∞) norm* T can be written as $p(y) = \sum_{\substack{j_1,\ldots,j_m \geq 0 \\ j_1+\cdots+j_m \leq t}} c_{j_1,\ldots,j_m} \prod_{\ell=1}^m y_\ell^{j_\ell}$ where $|c_{j_1,\ldots,j_m}| \leq T$ for every $j_1, \ldots, j_m$. Recall that there are at most $\binom{m+t}{t}$ coefficients in an m-variate polynomial of total degree t. Often we will want to associate a polynomial p of degree t and norm T with its coefficient vector $\boldsymbol{p} \in [-T, T]^{\binom{m+t}{t}}$. Specifically, $\boldsymbol{p} = (c_{j_1,\ldots,j_m})_{\substack{j_1,\ldots,j_m \geq 0 \\ j_1+\cdots+j_m \leq t}}$. Given a vector $\boldsymbol{p}$ and a point $y \in \{0,1\}^m$ we use $\boldsymbol{p}(y)$

to indicate the evaluation of the polynomial described by the vector $\boldsymbol{p}$ at the point y. Observe this is equivalent to computing $\boldsymbol{p} \cdot \boldsymbol{y}$ where $\boldsymbol{y} \in \{0,1\}^{\binom{m+t}{t}}$ is defined as $y_{j_1,\ldots,j_m} = \prod_{\ell=1}^{m} y_\ell^{j_\ell}$ for every $j_1, \ldots, j_m \geq 0$, $j_1 + \cdots + j_m \leq t$.

Let $\mathcal{P}_{t,T}$ be the family of all m-variate real polynomials of degree t and norm T. In many cases, the functions $f_{\mathcal{Q},x} : \{0,1\}^m \to \{0,1\}$ can be approximated well on all the indices in $Y_{\mathcal{Q}}$ by a family of polynomials $\mathcal{P}_{t,T}$ with low degree and small norm. Formally:

Definition 8 (Uniform Approximation by Polynomials). *Given a family of m-variate functions $\mathcal{F} = \{f_x\}_{x \in \mathcal{X}}$ and a set $Y \subseteq \{0,1\}^m$, we say that the family $\mathcal{P}_{t,T}$* uniformly γ-approximates *$\mathcal{F}$ on Y if for every $x \in \mathcal{X}$, there exists $p_x \in \mathcal{P}_{t,T}$ such that $\max_{y \in Y} |f_x(y) - p_x(y)| \leq \gamma$.*

We say that $\mathcal{P}_{t,T}$ efficiently and uniformly γ-approximates *$\mathcal{F}$ if there is an algorithm $\mathcal{P}_{\mathcal{F}}$ that takes $x \in \mathcal{X}$ as input, runs in time $\mathrm{poly}(\log |\mathcal{X}|, \binom{m+t}{t}, \log T)$, and outputs a coefficient vector $\boldsymbol{p}_x$ such that $\max_{y \in Y} |f_x(y) - \boldsymbol{p}_x(y)| \leq \gamma$.*

3 From Polynomial Approximations to Data Release Algorithms

In this section we present an algorithm for privately releasing any family of counting queries $\mathcal{Q}$ such that $\mathcal{F}_{\mathcal{Q}}$ that can be efficiently and uniformly approximated by polynomials. The algorithm will take an n-row database D and, for each row $x \in D$, constructs a polynomial p_x that uniformly approximates the function $f_{\mathcal{Q},x}$ (recall that $f_{\mathcal{Q},x}(q) = q(x)$, for each $q \in \mathcal{Q}$). From these, it constructs a polynomial $p_D = \frac{1}{n} \sum_{x \in D} p_x$ that uniformly approximates $f_{\mathcal{Q},D}$. The final step is to perturb each of the coefficients of p_D using noise from a Laplace distribution (Theorem 6) and bound the error introduced from the perturbation.

Theorem 9 (Releasing Polynomials). *Let $\mathcal{Q} = \{q_y\}_{y \in Y_{\mathcal{Q}} \subseteq \{0,1\}^m}$ be a set of counting queries over $\{0,1\}^d$, and $\mathcal{F}_{\mathcal{Q}}$ be the $\mathcal{Q}$ function family (Definition 7). Assume that $\mathcal{P}_{t,T}$ efficiently and uniformly γ-approximates $\mathcal{F}_{\mathcal{Q}}$ on $Y_{\mathcal{Q}}$ (Definition 8). Then there is a sanitizer $\mathcal{A}\colon (\{0,1\}^d)^n \to \mathbb{R}^{\binom{m+t}{t}}$ that*

1. *is ε-differentially private,*
2. *runs in time $\mathrm{poly}(n, d, \binom{m+t}{t}, \log T, \log(1/\varepsilon))$, and*
3. *is (α, β)-accurate for $\mathcal{Q}$ for $\alpha = \gamma + \frac{4T\binom{m+t}{t}^2 \log\left(\binom{m+t}{t}/\beta\right)}{\varepsilon n}$.*

Proof. First we construct the sanitizer $\mathcal{A}$. See the relevant codebox below.

Privacy. We establish that $\mathcal{A}$ is ε-differentially private. This follows from the observation that for any two adjacent $D \sim D'$ that differ only on row i^*,

$$\|\boldsymbol{p}_D - \boldsymbol{p}_{D'}\|_\infty = \left\| \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{p}_{x^{(i)}} - \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{p}_{x'^{(i)}} \right\|_\infty = \frac{1}{n} \left\| \boldsymbol{p}_{x^{(i^*)}} - \boldsymbol{p}_{x'^{(i^*)}} \right\|_\infty \leq \frac{2T}{n}.$$

The Sanitizer $\mathcal{A}$

Input: A database $D \in (\{0,1\}^d)^n$, an explicit family of polynomials $\mathcal{P}$, and a parameter $\varepsilon > 0$.

For $i = 1, \ldots, n$

Using efficient approximation of $\mathcal{F}$ by $\mathcal{P}$, compute a polynomial $\boldsymbol{p}_{x^{(i)}} = \mathcal{P}_{\mathcal{F}}(x^{(i)})$ that γ-approximates $f_{x^{(i)}}$ on $Y_{\mathcal{Q}}$.

Let $\boldsymbol{p}_D = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_{x^{(i)}}$, where the sum denotes standard entry-wise vector addition.

Let $\tilde{\boldsymbol{p}}_D = \boldsymbol{p}_D + Z$, where Z is drawn from an $\binom{m+t}{t}$-variate Laplace distribution with parameter $2T/\varepsilon n$ (Section 2.1).

Output: $\tilde{p}_D$.

The last inequality is from the fact that for every x, $\boldsymbol{p}_x$ is a vector of L_∞ norm at most T. Part 1 of the Theorem now follows directly from the properties of the Laplace Mechanism (Theorem 6). Now we construct the evaluator $\mathcal{E}$.

The Evaluator $\mathcal{E}$ for the Sanitizer $\mathcal{A}$

Input: A vector $\tilde{\boldsymbol{p}} \in \mathbb{R}^{\binom{m+t}{t}}$ and the description of a query $y \in \{0,1\}^m$.

Output: $\tilde{\boldsymbol{p}}(y)$. Recall that we view $\tilde{\boldsymbol{p}}$ as an m-variate polynomial, p, and $\tilde{\boldsymbol{p}}(y)$ is the evaluation of p on the point y.

Efficiency. Next, we show that $\mathcal{A}$ runs in time $\mathrm{poly}(n, d, \binom{m+t}{t}, \log T, \log(1/\varepsilon))$. Recall that we assumed the polynomial construction algorithm $\mathcal{P}$ runs in time $\mathrm{poly}(d, \binom{m+t}{t}, \log T)$. The algorithm $\mathcal{A}$ needs to run $\mathcal{P}_{\mathcal{F}}$ on each of the n rows, and then it needs to generate $\binom{m+t}{t}$ samples from a univariate Laplace distribution with magnitude $\mathrm{poly}(T, \binom{m+t}{t}, 1/n, 1/\varepsilon)$, which can also be done in time $\mathrm{poly}(\binom{m+t}{t}, \log T, \log n, \log(1/\varepsilon))$. We also establish that $\mathcal{E}$ runs in time $\mathrm{poly}(\binom{m+t}{t}, \log T, \log n, \log(1/\varepsilon))$, observe that $\mathcal{E}$ needs to expand the input into an appropriate vector of dimension $\binom{m+t}{t}$ and take the inner product with the vector $\tilde{\boldsymbol{p}}$, whose entries have magnitude $\mathrm{poly}(\binom{m+t}{t}, T, 1/n, 1/\varepsilon)$. These observations establish Part 2 of the Theorem.

Accuracy. Finally, we analyze the accuracy of the sanitizer $\mathcal{A}$. First, by the assumption that $\mathcal{P}_{t,T}$ uniformly γ-approximates $\mathcal{F}$ on $Y \subseteq \{0,1\}^m$, we have

$$\max_{y \in Y} |f_D(y) - \boldsymbol{p}_D(y)| \leq \frac{1}{n}\sum_{i=1}^{n} \max_{y \in Y} |f_{x^{(i)}}(y) - \boldsymbol{p}_{x^{(i)}}(y)| \leq \gamma.$$

Now we want to establish that $\Pr\left[\max_{y \in \{0,1\}^m} |\tilde{\boldsymbol{p}}_D(y) - \boldsymbol{p}_D(y)| \leq \alpha'\right] \geq 1 - \beta$ for $\alpha' = 4T\binom{m+t}{t}^2 \log\left(\binom{m+t}{t}/\beta\right)/\varepsilon n$, where the probability is taken over the coins of $\mathcal{A}$. Part (3) of the Theorem will then follow by the triangle inequality.

To see that the above statement is true, observe that by the properties of the Laplace mechanism (Theorem 6), we have $\Pr\left[\|\tilde{\boldsymbol{p}}_D - \boldsymbol{p}_D\|_1 \leq \alpha'\right] \geq 1 - \beta$, where the probability is taken over the coins of $\mathcal{A}$. Given that $\|\tilde{\boldsymbol{p}}_D - \boldsymbol{p}_D\|_1 \leq \alpha'$, it

holds that for every $y \in \{0,1\}^m$,

$$|\tilde{\boldsymbol{p}}_D(y) - \boldsymbol{p}_D(y)| = |(\tilde{\boldsymbol{p}}_D - \boldsymbol{p}_D)(y)| \leq \|\tilde{\boldsymbol{p}}_D - \boldsymbol{p}_D\|_1 \leq \alpha'.$$

The first inequality follows from the fact that every monomial evaluates to 0 or 1 at the point y. This completes the proof of the theorem.

4 Applications

In this section we establish the existence of explicit families of low-degree polynomials approximating the families $\mathcal{F}_{\mathcal{Q}}$ for some interesting query sets.

4.1 Releasing Monotone Disjunctions

We define the class of monotone k-way disjunctions as follows:

Definition 10 (Monotone k-Way Disjunctions). *Let $\mathcal{X} = \{0,1\}^d$. The query set $\mathcal{Q}_{\mathrm{Disj},k} = \{q_y\}_{y \in Y_k \subseteq \{0,1\}^d}$ of* monotone k-way disjunctions over $\{0,1\}^d$ *contains a query q_y for every $y \in Y_k = \left\{y \in \{0,1\}^d \mid |y| \leq k\right\}$. Each query is defined as $q_y(x_1, \ldots, x_d) = \bigvee_{j=1}^d y_j x_j$. The $\mathcal{Q}_{\mathrm{Disj},k}$ function family $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}} = \{f_x\}_{x \in \{0,1\}^d}$ contains a function $f_x(y_1, \ldots, y_d) = \bigvee_{j=1}^d y_j x_j$ for every $x \in \{0,1\}^d$.*

Thus the family $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}}$ consists of all disjunctions, and the index set, Y_k, consists of all vectors $y \in \{0,1\}^d$ with at most k non-zero entries.

The next lemma shows that $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}}$ can be efficiently and uniformly approximated by polynomials of low degree and low norm. The statement is a well-known application of Chebyshev polynomials, and a similar statement appears in [14] but without bounding the running time of the construction or a bound on the norm of the polynomials.

Lemma 11 (Approximating $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}}$ by polynomials, similar to [14]). *For every $k, d \in \mathbb{N}$ such that $k \leq d$ and every $\gamma > 0$, the family $\mathcal{P}_{t,T}$ of d-variate real polynomials of degree $t = O(\sqrt{k}\log(1/\gamma))$ and norm $T = d^{O(\sqrt{k}\log(1/\gamma))}$ efficiently and uniformly γ-approximates the family $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}}$ on the set Y_k.*

Theorem 1 in the introduction follows by combining Theorems 9 and 11.

4.2 Releasing Monotone r-of-k Queries

We define the class of monotone r-of-k queries as follows:

Definition 12 (Monotone r-of-k Queries). *Let $\mathcal{X} = \{0,1\}^d$ and $r, k \in \mathbb{N}$ such that $r \leq k \leq d$. The query set $\mathcal{Q}_{r,k} = \{q_y\}_{y \in Y_k \subseteq \{0,1\}^d}$ of* monotone r-of-k queries over $\{0,1\}^d$ *contains a query q_y for every $y \in Y_k = \left\{y \in \{0,1\}^d \mid |y| \leq k\right\}$. Each query is defined as $q_y(x_1, \ldots, x_d) = \mathbf{1}_{\sum_{j=1}^d y_j x_j \geq r}$. The $\mathcal{Q}_{r,k}$ function family $\mathcal{F}_{\mathcal{Q}_{r,k}} = \{f_x\}_{x \in \{0,1\}^d}$ contains a function $f_x(y_1, \ldots, y_d) = \mathbf{1}_{\sum_{j=1}^d y_j x_j \geq r}$ for every $x \in \{0,1\}^d$.*

The next lemma shows that $\mathcal{F}_{\mathcal{Q}_{r,k}}$ can be efficiently and uniformly approximated over Y_k by low-degree polynomials. The statement is based on approximation-theoretic results of Sherstov [18, Lemma 3.11].

Lemma 13 (Approximating $\mathcal{F}_{\mathcal{Q}_{r,k}}$ on Y_k). *For every $r, k, d \in \mathbb{N}$ such that $r \leq k \leq d$ and every $\gamma > 0$, the family $\mathcal{P}_{t,T}$ of d-variate real polynomials of degree $t = \tilde{O}(\sqrt{kr\log(1/\gamma)})$ and norm $T = d^{\tilde{O}(\sqrt{kr\log(1/\gamma)})}$ efficiently and uniformly γ-approximates the family $\mathcal{F}_{\mathcal{Q}_{r,k}}$ on the set Y_k.*

Remark 1. Using the principle of inclusion-exclusion, the answer to a monotone r-of-k query can be written as a linear combination of the answers to $k^{O(r)}$ monotone k-way disjunctions. Thus, a sanitizer that is $(\alpha/k^{O(r)}, \beta)$-accurate for monotone k-way disjunctions implies a sanitizer that is (α, β)-accurate for monotone r-of-k queries. However, combining this implication with Theorem 1 yields a sanitizer with running time $d^{O(r\sqrt{k}\log(k/\beta))}$, which has a worse dependence on r than what we achieve in Theorem 2.

4.3 Releasing Decision Lists

A *length-k decision list* over m variables is a function which can be written in the form "if ℓ_1 then output b_1 else $\cdots$ else if ℓ_k then output b_k else output b_{k+1}," where each ℓ_i is a boolean literal in $\{x_1, \ldots, x_m\}$, and each b_i is an output bit in $\{0, 1\}$. Note that decision lists of length-k strictly generalize k-way disjunctions and conjunctions. We use $\mathrm{DL}_{k,m}$ to denote the set of all length-k decision lists over m binary input variables.

Definition 14 (Evaluations of Length-k Decision Lists). *Let $k, m \in \mathbb{N}$ such that $k \leq m$ and $\mathcal{X} = \mathrm{DL}_{k,m}$. The query set $\mathcal{Q}_{\mathrm{DL}_{k,m}} = \{q_y\}_{y\in\{0,1\}^m}$ of* evaluations of length-k decision lists *contains a query q_y for every $y \in \{0, 1\}^m$. Each query is defined as $q_y(x) = x(y)$ where $x \in \mathrm{DL}_{k,m}$ is a length-k decision list over m variables. The $\mathcal{Q}_{\mathrm{DL}_{k,m}}$ function family $\mathcal{F}_{\mathcal{Q}_{\mathrm{DL}_{k,m}}} = \{f_x\}_{x\in\mathrm{DL}_{k,m}}$ contains functions $f_x(y) = x(y)$ for every $x \in \mathrm{DL}_{k,m}$. That is, $\mathcal{F}_{\mathcal{Q}_{\mathrm{DL}_{k,m}}} = \mathrm{DL}_{k,m}$.*

We clarify that in this setting, the records in the database are length-k decision lists over $\{0, 1\}^m$ and the queries inputs in $\{0, 1\}^m$. Thus $|\mathcal{X}| = |\mathrm{DL}_{k,m}| = m^{O(k)}$ and $|\mathcal{Q}| = 2^m$. Alternatively, $\mathcal{X} = \{0, 1\}^d$ for $d = k(\log m + 2) + 1$, since a length-k decision list can be described using this many bits.

Lemma 15 ([16]). *For every $k, m \in \mathbb{N}$ such that $k \leq m$ and every $\gamma > 0$, the family $\mathcal{P}_{t,T}$ of m-variate real polynomials of degree $\tilde{O}\left(\sqrt{k}\log(1/\gamma)\right)$ and norm $T = m^{\tilde{O}\left(\sqrt{k}\log(1/\gamma)\right)}$ efficiently and uniformly γ-approximates the family $\mathcal{F}_{\mathcal{Q}_{\mathrm{DL}_{k,m}}} = \mathrm{DL}_{k,m}$ on all of $\{0, 1\}^m$.*

We obtain Theorem 3 of the introduction by combining Theorems 9 and 15.

Acknowledgements

We thank Moritz Hardt, Varun Kanade, Aaron Roth, Guy Rothblum, and Li-Yang Tan for helpful discussions.

References

1. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., Talwar, K.: Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In: Libkin, L. (ed.) PODS. pp. 273–282. ACM (2007)
2. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: the SuLQ framework. In: Li, C. (ed.) PODS. pp. 128–138. ACM (2005)
3. Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: Dwork, C. (ed.) STOC. pp. 609–618. ACM (2008)
4. Cheraghchi, M., Klivans, A., Kothari, P., Lee, H.K.: Submodular functions are noise stable. In: Randall, D. (ed.) SODA. pp. 1586–1592. SIAM (2012)
5. Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: PODS. pp. 202–210. ACM (2003)
6. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: TCC '06. pp. 265–284 (2006)
7. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.P.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: STOC '09. pp. 381–390 (2009)
8. Dwork, C., Nissim, K.: Privacy-preserving datamining on vertically partitioned databases. In: Franklin, M.K. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 3152, pp. 528–544. Springer (2004)
9. Dwork, C., Rothblum, G.N., Vadhan, S.P.: Boosting and differential privacy. In: FOCS. pp. 51–60. IEEE Computer Society (2010)
10. Gupta, A., Hardt, M., Roth, A., Ullman, J.: Privately releasing conjunctions and the statistical query barrier. In: STOC '11. pp. 803–812 (2011)
11. Gupta, A., Roth, A., Ullman, J.: Iterative constructions and private data release. In: Cramer, R. (ed.) TCC. Lecture Notes in Computer Science, vol. 7194, pp. 339–356. Springer (2012)
12. Hardt, M., Ligett, K., McSherry, F.: A simple and practical algorithm for differentially private data release. CoRR abs/1012.4763 (2010)
13. Hardt, M., Rothblum, G.N.: A multiplicative weights mechanism for privacy-preserving data analysis. In: FOCS. pp. 61–70. IEEE Computer Society (2010)
14. Hardt, M., Rothblum, G.N., Servedio, R.A.: Private data release via learning thresholds. In: Randall, D. (ed.) SODA. pp. 168–187. SIAM (2012)
15. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? SIAM J. Comput. 40(3), 793–826 (2011)
16. Klivans, A.R., Servedio, R.A.: Toward attribute efficient learning of decision lists and parities. In: Shawe-Taylor, J., Singer, Y. (eds.) COLT. Lecture Notes in Computer Science, vol. 3120, pp. 224–238. Springer (2004)
17. Roth, A., Roughgarden, T.: Interactive privacy via the median mechanism. In: STOC '10. pp. 765–774 (2010)
18. Sherstov, A.A.: Approximate inclusion-exclusion for arbitrary symmetric functions. Computational Complexity 18(2), 219–247 (2009)
19. Thaler, J., Ullman, J., Vadhan, S.: Faster algorithms for privately releasing marginals. CoRR abs/1205.1758 (2012)
20. Ullman, J., Vadhan, S.P.: PCPs and the hardness of generating private synthetic data. In: TCC '11. pp. 400–416 (2011)