On the Complexity of Differentially Private Data Release

Efficient Algorithms and Hardness Results

Cynthia Dwork [*]
Microsoft Research SVC

Moni Naor [†]
Weizmann Institute

Omer Reingold [‡]
Weizmann Institute

Guy N. Rothblum [§]
MIT

Salil Vadhan [¶]
Harvard University

ABSTRACT

We consider private data analysis in the setting in which a trusted and trustworthy curator, having obtained a large data set containing private information, releases to the public a "sanitization" of the data set that simultaneously protects the privacy of the individual contributors of data and offers utility to the data analyst. The sanitization may be in the form of an arbitrary data structure, accompanied by a computational procedure for determining approximate answers to queries on the original data set, or it may be a "synthetic data set" consisting of data items drawn from the same universe as items in the original data set; queries are carried out as if the synthetic data set were the actual input. In either case the process is non-interactive; once the sanitization has been released the original data and the curator play no further role.

For the task of sanitizing with a synthetic dataset output, we map the boundary between computational feasibility and infeasibility with respect to a variety of utility measures. For the (potentially easier) task of sanitizing with unrestricted output format, we show a tight qualitative and quantitative connection between hardness of sanitizing and the existence of traitor tracing schemes.

Categories and Subject Descriptors

F.2.0 [**Theory of Computation**]: Analysis of Algorithms and problem complexity—*General*

General Terms

Algorithms, Security, Theory

1. INTRODUCTION

We consider private data analysis in the setting in which a trusted and trustworthy curator, having obtained a large data set containing private information, releases to the public a "sanitization" of the data set that simultaneously protects the privacy of the individual contributors of data and offers utility to the data analyst. The literature refers to this as the *non-interactive* model, since once the curator has released the sanitization there is no further use for either the original data or the curator.

There has been a series of negative results concerning privacy, roughly saying that there is a class of queries with the property that it is blatantly non-private (allowing almost full reconstruction) if "too many" queries receive "overly accurate" responses [5, 7, 9]. These results have been interpreted to mean that one cannot answer a linear, in the database size, number of queries with small noise while preserving privacy. This view motivates an interactive approach to private data analysis where the number of queries is limited to be small — sublinear in the size n of the dataset. Intuitively, in the interactive approach only the questions actually asked receive responses, while in the non-interactive approach, if there is no way to anticipate what will be of interest to the analyst, all questions must receive relatively accurate responses, which, by the aforementioned results, leads to blatant non-privacy.

Against this backdrop, Blum, Ligett and Roth revisited the non-interactive case from a learning theory perspective [1] and contradicted the above interpretation about the necessity of limiting the number of queries to be sublinear. Let X be a universe of data items and $\mathcal{C}$ be a "concept" class consisting of efficiently computable functions $c : X \to \{0, 1\}$.

[*]E-mail: dwork@microsoft.com.

[†]Incumbent of the Judith Kleeman Professorial Chair, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: moni.naor@weizmann.ac.il. Research supported in part by a grant from the Israel Science Foundation.

[‡]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: omer.reingold@weizmann.ac.il. Work done in part while visiting Microsoft Research Silicon Valley. Research supported by grant 1300/05 from the Israel Science Foundation.

[§]CSAIL, MIT. E-mail: rothblum@csail.mit.edu. Most work done while at Microsoft Research, also supported by NSF Grants CCF-0635297, NSF-0729011, CNS-0430336.

[¶]School of Engineering and Applied Sciences and Center for Research on Computation and Society, Harvard University. E-mail: salil@eecs.harvard.edu. Work begun while visiting Microsoft Research Silicon Valley. Also supported by NSF grant CNS-0831289.

STOC'09, May 31–June 2, 2009, Bethesda, Maryland, USA.

Given a database $D \in X^n$, Blum *et al.* employ the exponential mechanism of McSherry and Talwar [17] to (inefficiently) obtain a *synthetic database* that remarkably maintains approximately correct fractional counts for *all* concepts in $\mathcal{C}$ simultaneously, while ensuring a very strong privacy guarantee. That is, letting y denote the synthetic database produced, with high probability over the choices made by the curator, for every concept $c \in \mathcal{C}$, the fraction of elements in y that satisfy c is approximately the same as the fraction of elements in D that satisfy c.[1]

While the Blum *et al.* result is striking in its generality and its privacy/utility tradeoff, a direct implementation of the mechanism takes time superpolynomial in $|X|$ and $|\mathcal{C}|$, even though objects being manipulated are of bit-length only $\text{poly}(n, \log|X|, \log|\mathcal{C}|)$. In this paper, we investigate the computational feasibility of non-interactive privacy mechanisms, delineating the boundary between efficient and inefficient sanitization.

Synthetic Databases. Very roughly, we show that if either the universe X of data items or the concept class $\mathcal{C}$ is of size superpolynomial in a computation parameter κ, then there exists a distribution on databases and a concept class $\mathcal{C}$ for which there is no efficient (in κ) mechanism for privately generating synthetic databases. It follows that there is no general efficient implementation of the exponential mechanism. In contrast, if both the concept class and the data universe are of size polynomial in κ, then not only is there an efficient mechanism, but the size of the input database can be surprisingly small, namely $O(2^{\sqrt{\log|\mathcal{C}|}}\log|X|)$, that is $|\mathcal{C}|$ may be as large as $n^{O(\log n)}$.

Non-synthetic case. What about the seemingly easier problem of efficiently and privately generating a *data structure* that for each $c \in \mathcal{C}$, yields an approximation to the fraction of items in the database that satisfy c? Here we show a tight connection between hardness of sanitization and the existence of *traitor tracing* schemes in cryptography [4]. Thus we have here a dichotomy: the good news of one area (tracing traitors) is the bad news of the other (privacy preserving data release). This is somewhat similar to, but actually sharper than, the dichotomy between the existence pseudo-random functions and (hardness of) learning general functions[20].

2. PRELIMINARIES AND DEFINITIONS

Let $[n]$ be the set $\{1, 2, \ldots n\}$. For $x, y \in \{0,1\}^n$ we use $x \circ y$ to denote the concatenation of x and y (a string in $\{0,1\}^{2n}$). For a (discrete) distribution $\mathcal{D}$ over a set X we denote by $x \sim \mathcal{D}$ the experiment of selecting $x \in X$ by the distribution $\mathcal{D}$. A function $f(n)$ is *negligible* if it is smaller than any (inverse) polynomial. We let X denote the universe of data items (database rows), $\mathcal{C}$ denote the concept class, and $\mathcal{D}$ denote a distribution on databases. The (randomized) non-interactive privacy mechanism is denoted A (we sometimes refer to A as a sanitizer); typically A will operate on an input database $D \in X^n$. For a given database D, the sanitizer A computes an output $A(D)$ that can later be used to reconstruct information about D. When A has synthetic database output, the output of $A(D)$ is itself a synthetic database $y \in X^m$. *Neighboring* databases are databases of symmetric difference 1.

[1]This does not contradict the negative results because of the size of the error in the case of attacks using a polynomial number of queries, or the size of the input database in the case of attacks using an exponential number of queries.

2.1 Privacy Preserving Sanitizers

DEFINITION 2.1 (DIFFERENTIAL PRIVACY [6]). *A randomized function A is* (ε, δ)-differentially private *if for any pair (D, D') of neighboring databases and for all sets S of possible outputs of A:* $\Pr[A(D) \in S] \leq e^{\varepsilon} \cdot Pr[A(D') \in S] + \delta$. *If A is $(\varepsilon, 0)$-differentially private (i.e. $\delta = 0$), then we say it is* ε-differentially private.

The *sensitivity* of a function f is the maximum, over all paris of neighboring databases D and D', in the difference of f's values $|f(D) - f(D')|$. The *Laplace distribution* $Lap(t)$ has density function $h(y) \propto e^{-y/t}$, has mean 0 and standard deviation $\sqrt{2}t$. We usually refer to the Laplace distribution over integers. Dwork *et al.* [6] showed that if a function f has global sensitivity s then the function $f(D) + Lap(s/\varepsilon)$ is ε-differentially private.

The *true answer* to a query $c \in \mathcal{C}$ on a database D is the number (or fraction) of elements in D that satisfy the predicate c. For $c \in \mathcal{C}$ we say that $A(D)$ is α-*accurate for* c if the difference between the fractional count that $A(D)$ gives for c and the fractional count D gives for c (sometimes denoted $c(D)$) is at most α. We say that $A(D)$ is (α, γ)-*accurate for a concept class* $\mathcal{C}$ if $A(D)$ is α-accurate for a $1 - \gamma$ fraction of the concepts in $\mathcal{C}$ (if $\gamma = 0$ we sometimes refer to α-accuracy of $A(D)$ for a class). Our notion of a sanitizer A's utility w.r.t. a concept class is below.

DEFINITION 2.2 ((α, β, γ)-UTILITY). *Let $\mathcal{C}$ be a concept class and X a data universe. A sanitizer A has (α, β, γ)-utility for n-item databases w.r.t. $\mathcal{C}$ and X if for any n-item database D:* $\Pr_{A\text{'s coins}}[A(D)$ *is* (α, γ)-*accurate*$] \leq \beta$

We are mostly interested in coming up with sanitizers with $\gamma = 0$, however both as an intermediate result and as an exploration of the limitation of hardness we will consider the general case.

2.2 Hardness of Sanitizing

In this section we define what it means for a distribution on databases to be *computationally hard* to sanitize. Intuitively, for a hardness result to be convincing, we would like to say that it is hard to meet even a *weak* notion of privacy (let alone a notion as strong as differential privacy). In fact, we will say that sanitizing is hard if it is hard even to avoid leaking input items in their entirety: i.e. some item's privacy is always blatantly violated (such leakage is a very significant breach of privacy). Note that if leaking a few input items is allowed then sanitizing (with synthetic output) becomes easy: just output a randomly chosen subset of the input items; with high probably this subset will preserve utility even with respect to large sets of counting queries.

A distribution of databases is *hard to sanitize* (with respect to some concept class) if for any efficient alleged sanitizer, with high probability over a database drawn from the distribution, one of the database items can be extracted from the alleged sanitizer's output. To avoid triviality, we will also require that when this leaked item is excluded from the input database (and, say, replaced by a random different item), the probability that it can be extracted from the

output is very small. This means that any efficient (alleged) sanitizer indeed compromises the privacy of input items in a strong sense. All of our negative results imply such strong privacy breaches. A formal definition follows.

DEFINITION 2.3. *($(\mu, \alpha, \beta, \gamma, \mathcal{C})$ Hard to Sanitize Database Distribution) Let $\mathcal{C}$ be a concept class ensemble, X a data universe ensemble, and $\mu, \alpha, \beta, \gamma \in [0,1]$. Let n be a database size and $\mathcal{D}$ an ensemble of distributions, where $\mathcal{D}_n$ is over collections of $n+1$ items from X_n, we often denote by $(D, D_i') \sim \mathcal{D}$ the experiment of choosing a database of n elements, and an additional element D_i' from the distribution $\mathcal{D}$. We think of $\mathcal{D}$ as specifying a distribution on n-item databases (and their neighbors) that is hard to sanitize.*

An algorithm is said to be "efficient" if its running time is $\text{poly}(n, \log(|\mathcal{C}_n|), \log(|X_n|))$.[2] *We say that $\mathcal{D}$ is* $(\mu, \alpha, \beta, \gamma, \mathcal{C})$-hard-to-sanitize *if the following holds:*

There exists an efficient algorithm T such that for any alleged efficient sanitizer A the following two conditions hold:

1. *With probability $1-\mu$ over choosing a database D from $\mathcal{D}$ and over A's and T's coins, if $A(D)$ maintains α-utility for a $1-\gamma$ fraction of concepts, then T can recover one of D's items from $A(D)$: the probability (over $(D, D_i') \sim \mathcal{D}$, and over A's and T's coins) that ($A(D)$ maintains (α, γ)-utility) and $(D \cap T(A(D)) = \emptyset)$] is at most μ.*

2. *For* every *efficient algorithm A, and for every $i \in [n]$, if we draw (D, D_i') from $\mathcal{D}$, and replace D_i with D_i' to form D', T cannot extract D_i from $A(D')$ except with small probability:*

$$\Pr_{(D,D_i') \sim \mathcal{D}, A\text{'s, } T\text{'s coins}}[D_i \in T(A(D'))] \leq \mu$$

We often drop $\mathcal{C}$ when it is clear from the context, and refer to distributions that are $(\mu, \alpha, \beta, \gamma)$-hard to sanitize. In most of our examples μ will be a negligible function, we use (α, β, γ)-hard to sanitize as shorthand for $(neg(), \alpha, \beta, \gamma)$-hard to sanitize ($\mathcal{C}$ will be clear from the context).

We conclude this section by observing that in particular, if a distribution is hard to sanitize as in 2.3, then it is also hard to sanitize while achieving even weak differential privacy. In other words hard to sanitize implies hard to sanitize with differential privacy (the other implication is not true).

CLAIM 2.1. *If a database distribution $\mathcal{D}$ is hard to sanitize with respect to $(\mu, \alpha, \beta, \gamma, \mathcal{C})$, where $\mu \leq \min(\beta, (1-8\beta)/(8n^{1+a}))$ for some $a > 0$, then no efficient sanitizer that achieves (α, β, γ)-utility with respect to $\mathcal{C}$ on databases drawn from $\mathcal{D}$ can achieve $(a\log(n), (1-8\beta)/2n^{1+a})$ differential privacy.*

3. HARDNESS: SYNTHETIC DB OUTPUT

We use cryptography to obtain our negative results. Assume the existence of an existentially unforgeable signature scheme in which, given a set of signatures under a given key, it is infeasible to generate a new signature, either of one of the messages in the set or of any other message (known as strong unforgeability or super-secure, see Section 6.5.2 in [11]). The class $\mathcal{C}$ for which it is hard to come up with privacy-preserving synthetic data contains one concept for each verification key vk. The data universe X consists of the set of all possible (message, signature) pairs. Assume messages and signatures have length polynomial in κ. A data item (m, σ) belongs to the concept c if $Verify(vk, m, \sigma) = 1$, *i.e.*, if σ is a valid signature for m according to verification key vk. Each database in $\mathcal{D}$ will be a set of valid (message, signature) pairs, where all signatures are under the *same* signing key. Let $D \in_R \mathcal{D}$ be a database, and let s be the signing key used, with corresponding verification key vk. Assuming that the sanitizer has produced y, it must be the case that almost all rows of y are valid signatures under vk (because the fractional count of D for the concept vk is 1). By the strong unforgeability of the signature scheme all of these must come from the input database D, contradicting (any reasonable notion of) privacy.

In the example, both $\mathcal{C}$ (the set of verification keys) and X (the set of (message, signature) pairs) are large. In Theorems 3.1 and 3.2 we show that we can obtain hardness results even if *either* of these is large while the other has size polynomial in κ. In other words, there is no general method for efficient generation of privacy-preserving synthetic data, assuming the existence of one-way functions if $|\mathcal{C}|$ of $|X|$ is $\exp(\kappa)$. We note that, under stronger assumptions about one-way functions (e.g. sub-exponential time hardness), hardness results hold for much smaller $\mathcal{C}, X$.

THEOREM 3.1. *Let $f : \{0,1\}^\kappa \to \{0,1\}^\kappa$ be a one-way function. There exists a concept class $\mathcal{C}$ of size* $\text{poly}(\kappa)$*, a data universe X of size* $\exp(\text{poly}(\kappa))$*, and a distribution on databases of size* $\text{poly}(\kappa)$ *that is* $(\mu = neg(\kappa), \alpha, \beta, \gamma, \mathcal{C})$-hard-to-sanitize*, for any $4\gamma + 2\alpha \leq 1/4$ and any noticeable $1-\beta$.*

To come up with the hard to sanitize DB we start with the construction based on signatures outlined above. This time we have only a single concept that verifies the validity of signatures: the verification algorithm *without* the verification key hard-wired in. The verification key will now be included as part of the data items. We construct hard to sanitize databases by choosing a *single* verification key and generating multiple valid signed messages. Now, as in the case above, no sanitizer can output private synthetic data that uses the same verification key as the input database. However, since the verification key is no longer hardwired into the (single) concept, the sanitizer may just generate synthetic data with valid signatures under *a different key* of its choosing. To prevent the adversary from doing so, we add more concepts, so that preserving utility for these concepts forces the sanitizer not to change the verification key used in the input database. This can be done by adding concepts that output the bits of the encoding of the verification key, where the encoding is under a high-distance error correcting code.

THEOREM 3.2. *Let $f : \{0,1\}^\kappa \to \{0,1\}^\kappa$ be a one-way function. For every $a > 0$, and for every integer $n =$* $\text{poly}(\kappa)$*, there exists a concept class $\mathcal{C}$ of size* $\exp(\text{poly}(\kappa))$*, a data universe X of size $O(n^{2+2a})$, and a distribution on databases of size n that is* $(\mu, \alpha, \beta, 0, \mathcal{C})$- hard-to-sanitize *(i.e. hard to sanitize for worst-case concepts) for $\alpha \leq 1/3$, $\beta \leq 1/10$ and $\mu = 1/40n^{1+a}$.*

[2]More generally, we could also speak of hardness w.r.t. t-time sanitizers. For simplicity, we deal only with hardness w.r.t. $\text{poly}(n, \log(|\mathcal{C}_n|), \log(|X_n|))$-time sanitizers except when we note otherwise.

In this construction we will use pseudo-random functions [12]: a family of efficiently computable functions, such that a random function from the family is indistinguishable (via black-box access) from truly random functions. We will look at a family of functions with polynomial-size domain and range. The data items are input-output pairs and there is a concept for every function in the pseudo-random family that accepts an input-output pair iff it is consistent with the function. We generate a hard-to-sanitize database D by choosing a random function f_s in the family, n random inputs, computing the n outputs of f_s on those inputs, and putting the input-output pairs in the database. The intuition is that to maintain utility on the concept corresponding to f_s, most of the items in the output of a sanitizer with synthetic data should be input-output pairs consistent with f_s. But f_s is a pseudorandom function, and so, for inputs to f_s that were not in the initial database, the sanitizer's probability of "guessing" f_s's output is polynomially small. Thus (with high probability) many of the items that were in the input database must also be in the sanitizer's output.

4. POSITIVE RESULTS

4.1 Arbitrary Outputs to Synthetic Data

We begin by observing that a sanitizer with arbitrary output can be transformed into one with a synthetic datasets output. The main cost of the transformation is an additional running time and error overhead.

THEOREM 4.1. *Let X be a data universe, $\mathcal{C}$ a concept class and a A an (ε, δ)-differentially private sanitizer with utility $(\alpha, \beta, 0)$ and arbitrary output.*

Then there exists a sanitizer A' that is (ε, δ)-differentially private and has utility $(4\alpha, 2\beta, 0)$. The new sanitizer A' outputs a synthetic database of size $\tilde{O}(\log(|\mathcal{C}|)\cdot\log(1/\beta)/\alpha^2)$, its runtime is polynomial in A's and in $(|X|, |\mathcal{C}|, 1/\alpha, \log(1/\beta))$

PROOF. The idea is to run the sanitizer A and then use it to get (differentially private) counts on all the concepts in $\mathcal{C}$. We will then use linear programming to come up with a low-weight fractional database that approximates these counts. We transform this fractional database into a standard synthetic database by rounding the fractional counts.

The new sanitizer A' begins by running A on its input database D, and then for each concept $c \in \mathcal{C}$ it uses A's output to compute a fractional count on that concept. The input database D is never accessed again and so A' is (ε, δ)-differentially private. Let v be the resulting vector of counts, i.e. v_c is the fractional count that A's output gives on concept c. With probability $1 - \beta$, all of v's entries are α-accurate. We now want to extract a "fractional" database y that approximates these counts. This will be done using linear programming. For every $i \in X$ we introduce a variable $a_i \geq 0$ that will "count" the (fractional) number of occurrences of i in the fractional database y. We represent the count of concept c in y as the sum of the count of items that satisfy c. We want all of these counts to be within an α-accuracy of the respective counts in v_c, writing this as a linear inequality we get:

$$(v_c - \alpha) \cdot \sum_{i \in X} a_i \quad \leq \quad \sum_{i \in X \text{ s.t. } c(i)=1} a_i \leq (v_c + \alpha) \cdot \sum_{i \in X} a_i$$

When the counts are all within an α-accuracy of the counts in v_c, it is also the case that (with probability $1 - \beta$) they are all within a 2α-accuracy of the correct counts.

We write a linear program with two such constraint for each concept (a total of $2|\mathcal{C}|$ constraints). A' tries to find a fractional solution to this linear program. To see that such a solution exists, observe that the database D itself is α-close to the vector of counts v, and so there *exists* a solution to the linear program (in fact even an integer solution), and hence A' will find *some* fractional solution.

We conclude that A' can generate a fractional database with $(2\alpha, \beta, 0)$-utility, but we really want a synthetic (integer) database. To transform the fractional database into an integer one, we first scale the fractional database so that its total weight is 1, i.e. $\sum_{i \in X} a_i = 1$. Now round down each fractional point to closest multiple of $\alpha/|X|$, this changes each fractional count by at most a $\alpha/|X|$ additive factor, and so the rounded counts have $(3\alpha, \beta, 0)$ utility. Now we can treat the rounded fractional database (which has total weight 1), as an integer synthetic database of (polynomial) size at most $|X|/\alpha$. If this size is too large, we can reduce it by randomly sampling $m = \tilde{O}(\log(|\mathcal{C}|) \cdot \log(1/\beta)/\alpha^2)$ items. With probability $1 - \beta$ the resulting database maintains 4α-utility on all of the concepts. Taking a union bound (over the probability of A failing and the probability of the random sub-sampling failing) we get $(4\alpha, 2\beta, 0)$-utility. □

4.2 On-Average Utility

In this section we show how to transform a sanitizer with synthetic data that gives worst-case utility for any small set of concepts from a class $\mathcal{C}$ into a sanitizer that gives average-case utility for *the entire class* $\mathcal{C}$. This transformation works for any concept class $\mathcal{C}$ and data universe X as long as the output of the sanitizer is not too large.

THEOREM 4.2. *Let $\mathcal{C}$ be a concept class that can be efficiently sampled and X a data universe, and let A be an (ε, δ)-differentially private sanitizer that, given a subclass $\mathcal{C}' \subseteq \mathcal{C}$ of up to s concepts, achieves $(\alpha, \beta, 0)$-utility for all the concepts in $\mathcal{C}'$. Furthermore, A's output is a synthetic data set of total size m bits.*

Then there is an (ε, δ)-differentially private sanitizer A' that achieves $(\alpha, 2\beta, (m + \log(1/\beta))/s)$ utility for the class $\mathcal{C}$. The running time of A' is polynomial in A's, and it also outputs a synthetic data set of total size m bits.

PROOF. The sanitizer A' works by choosing *at random* a set $\mathcal{C}'$ of s concepts from $\mathcal{C}$ and running A on its input database D with the concepts in $\mathcal{C}'$, outputting a small m-bit sanitization $A(D)$. This sanitization is A''s output and so it is clearly (ε, δ)-differentially private.

For utility, we claim that with probability $1 - \beta$ over the selection of the concept set $\mathcal{C}'$ there *does not exist* an m-bit output sanitization y and a collection $\mathcal{C}_y$ of concepts of size $|\mathcal{C}|/s \cdot (m + 2\log(1/\beta))$ such that y is α-accurate for *all* of the concepts in $\mathcal{C}'$, but bad for all the concepts in $\mathcal{C}_y$. Once we prove this, we know that with high probability over $\mathcal{C}'$, whenever $A(D)$ is accurate for all of the concepts in $\mathcal{C}'$, it is also accurate for all but $|\mathcal{C}|/s \cdot (m + 2\log(1/\beta))$ of the concepts in $\mathcal{C}$. Taking a union bound, A''s total failure probability is at most 2β.

Examine a potential m-bit output y of A, and say that y is "bad" if it gives counts that are not α-accurate for a concept set $\mathcal{C}_y$ of total size $|\mathcal{C}| \cdot (\log(1/\beta) + m)/s$ (within $\mathcal{C}$). For such a bad y, what is the probability (over $\mathcal{C}'$) that y gives

accurate counts for *every* concept in $\mathcal{C}'$? This is exactly the probability that none of the concepts in $\mathcal{C}'$ are in $\mathcal{C}_y$, or

$$\begin{aligned}(1-|\mathcal{C}_y|/|\mathcal{C}|)^{|\mathcal{C}'|} &\leq (1-(\log(1/\beta)+m)/s)^s \\ &\leq e^{-(\log(1/\beta)+m)} \leq \beta\cdot 2^{-m}\end{aligned}$$

Now, taking a union bound, the probability there *exists* an m-bit output y accurate on the concepts in $\mathcal{C}'$ but inaccurate on a set of size $|\mathcal{C}|\cdot(\log(1/\beta)+m)/s$ is at most β. □

REMARK 4.1. *If, instead of producing a synthetic data set, the sanitizer A used in Theorem 4.2 produced an arbitrary data structure accurately covering $\mathcal{C}'$, we would not know how to argue that it also gives information about $\mathcal{C}\setminus\mathcal{C}'$. This is all we need, but we do not in general know how to interpret an arbitrary data structure for $\mathcal{C}'$ on concepts not in $\mathcal{C}'$. In contrast, an item from the data universe always either does or does not satisfy an arbitrary concept in $\mathcal{C}$.*

Note that if the sanitizer produces a synthetic database, then we can ensure the condition in Theorem 4.2 on the smallness of the output by subsampling, at a small cost in utility: for any sanitizer A with synthetic output and (α,β,γ)-utility, we can randomly sub-sample $\tilde{O}(\log(1/\beta)\cdot\log(|\mathcal{C}|)/\alpha^2)$ items. This gives us an m-bit output, where the value of m is $m=\tilde{O}(\log(|X|)\cdot\log(1/\beta)\cdot\log(|\mathcal{C}|)/\alpha^2)$. With probability $1-\beta$ the counts of the small subsampled synthetic database are α-close to those of A's output, and are thus 2α-accurate for all of the concepts A was accurate on. Thus we get a sanitizer with small output and $(2\alpha,2\beta,\gamma)$-utility and small output (there is no loss in privacy).

Average-Case Utility for small X. When X is not too large, by Theorem 4.1 we can transform any sanitizer A on small concept classes into one that outputs small synthetic databases. This means that if we permit sanitizers to run in time $\text{poly}(|X|)$, *any* worst-case sanitizer for small sets of concepts (with or without synthetic data) can be transformed into an average-case sanitizer for larger sets of concepts. Even for an exponential-size concept class $\mathcal{C}$, we can choose a polynomial size collection of random concepts $\mathcal{C}'$, efficiently sanitize w.r.t. $\mathcal{C}'$ (with a small synthetic data output) and get utility for all but a small polynomial fraction of the concepts in the exponential class $\mathcal{C}$.

4.3 Main Positive Result

In this section we construct an efficient sanitizer that has small error even when the concept class $\mathcal{C}$ is large:

THEOREM 4.3. *Let $\mathcal{C}$ be a concept class and X a data universe. Fix any desired $\varepsilon>0$ and a security parameter κ s.t. $\exp(\kappa)>|\mathcal{C}|$. There is an $(\varepsilon,\exp(-\kappa))$-differentially private sanitizer A with utility:*

$$(\tilde{O}(\frac{\kappa^5\cdot\sqrt{\log|X|}\cdot 2^{\sqrt{\log(|\mathcal{C}|)}}}{\varepsilon\cdot\sqrt{n}}),\exp(-\kappa))$$

A's runtime is $\text{poly}(n,|\mathcal{C}|,|X|,\kappa)$.

Construction Overview. The construction is recursive: we start out wanting to construct a sanitizer for a large concept class $\mathcal{C}$ of size $|\mathcal{C}|$. We reduce this to the goal of constructing a sanitizer for any subset $\mathcal{C}'\subseteq\mathcal{C}$ of size $|\mathcal{C}|/f$ where f is some factor by which we shrink the concept class. The reduction is stated as Lemma 4.4 below, and the intuition follows the proof of Theorem 4.2. Say we have a sanitizer for sets of $|\mathcal{C}|/f$ concepts with small synthetic output: we sample a set of $|\mathcal{C}|/f$ concepts uniformly from $\mathcal{C}$ and run this sanitizer on them. With overwhelming probability, the resulting synthetic database y gives accurate counts for all but a small set of concepts in $\mathcal{C}$. We release (in a privacy-preserving manner) the "names" of the concepts for which y does not give accurate counts, and we also release these concepts' counts in the input database D (adding noise to make sure these counts also preserve privacy). Releasing these "names" in a privacy-preserving manner is one of the main technical obstacles, as for each of the $|\mathcal{C}|$ concepts in the class we want to release (in a privacy-preserving manner) whether y gives accurate counts. This is a lot of information to release, essentially we are answering a query for each $c\in\mathcal{C}$, and the challenge is to answer all of these queries while maintaining privacy, see the details below.

The sanitizer for sets of $|\mathcal{C}|/f$ concepts used in the above reduction is constructed recursively via the same construction. The output size for the sanitizer constructed above is large (and it is not synthetic data), but we can reduce it to be a small synthetic output using Theorem 4.1. We do this recursively d times until in the end we only need to construct a sanitizer on sets of $|\mathcal{C}|/f^d$ concepts. In each of these recursive steps we pay additive losses in the privacy parameters and in β and a constant multiplicative factor in the α accuracy parameter (a factor of 4 comes from using Theorem 4.1 to generate synthetic data, in the actual construction there will be additional losses in accuracy). The recursion terminates once the concept class size is small enough to allow us to sanitize using a very simple sanitizer.

To get a better (but still rough) idea of the parameters, especially the dependence of the error on $|\mathcal{C}|$, observe that at the "bottom" of the recursion we have roughly $|\mathcal{C}|/f^d$ concepts remaining, where we need $|\mathcal{C}|/f^d>n$ to guarantee that a synthetic dataset that fits these concepts at the base of the recursion is $(1/\sqrt{n})$-fractionally accurate for the other concepts in the next recursion level up. We sanitize for these remaining concepts using the techniques of [5, 8] by outputting noisy counts for all the concepts (using binomial noise). This adds (non-fractional) error roughly proportional to $\sqrt{|\mathcal{C}|/f^d}$ (at least $\sqrt{n}$). In each of the recursive steps we are also adding noise at least $\sqrt{f}$ to sets of counts of size roughly f (actually more noise is being added). Finally, in each of the d levels of the recursion the error is blown up by a factor of 4 when we transform the output into a synthetic dataset. In total we get that the global (non-normalized) error is greater than roughly $4^d\cdot(\sqrt{|\mathcal{C}|/f^d}+\sqrt{f})$ (but not too much greater). To optimize this we choose $f=2^{\sqrt{\log(|\mathcal{C}|)}}$ and $d=\sqrt{\log(|\mathcal{C}|)}$, and get that the dependence of the error on $|\mathcal{C}|$ is roughly $2^{O(\sqrt{\log(|\mathcal{C}|)})}$, or $|\mathcal{C}|^{o(1)}$. We proceed to prove this result, starting with a Lemma that will be used to recurse in the construction.

LEMMA 4.4. *Let $\mathcal{C}$ be a concept class and X a data universe. Let A_{small} be an $(\varepsilon_{small},\delta_{small})$-differentially private sanitizer that, given any concept set $\mathcal{C}_{small}\subseteq\mathcal{C}$ of size s_{small}, achieves $(\alpha_{small},\beta_{small})$-utility on the concepts in $\mathcal{C}_{small}$. Say A_{small} has synthetic output of size m bits.*

Choose a desired ε and security parameter κ such that $\exp(\kappa)>|\mathcal{C}|$ and $(\beta_{small}+\exp(-\kappa))\leq min(1/2,\varepsilon/8)$. There exists a sanitizer A for the entire class $\mathcal{C}$. This sanitizer A is

$(\varepsilon_{small}+\varepsilon, \delta_{small}+\beta_{small}+\exp(-\kappa))$*-differentially private, it has utility:*

$$(\alpha_{small} + \frac{8\kappa^4 \cdot \sqrt{m}}{\varepsilon \cdot n} \cdot \sqrt{\frac{|\mathcal{C}|}{s_{small}}}, \beta_{small} + \exp(-\kappa))$$

The running time of the new sanitizer is $\text{poly}(n, |\mathcal{C}|, \log|X|, \kappa)$ *plus a single call to* A_{small}*.*

PROOF SKETCH. We choose a random subset $\mathcal{C}_{small}$ of s_{small} concepts in $\mathcal{C}$ and run the sanitizer A_{small} on this set with parameters $\varepsilon_{small}, \delta_{small}, \alpha, \beta$ (we need an input database of size n_{small}). Let y be the m-bit output of A_{small}.

As in Theorem 4.2, with overwhelming probability $1 - \exp(-\kappa)$ over the choice of $\mathcal{C}_{small}$, the set $\mathcal{C}_{bad}$ of concepts that y is not α-accurate on is of size smaller than

$$s_{bad} = |\mathcal{C}| \cdot (\kappa + m)/s_{small}$$

The sanitizer A outputs this (differentially private) y in the clear. There remains, however, the set $\mathcal{C}_{bad}$ of concepts for which y is not accurate. We would like to output which of the concepts are in $\mathcal{C}_{bad}$ and then separately release a differentially private sanitization that has good utility on this (much smaller) set of concepts. The problem is that the set of concepts that are in $\mathcal{C}_{bad}$ depends on the input database D, and so we must be careful to release it in a differentially private manner.

For each concept $c \in \mathcal{C}$ we want to release a bit v_c that indicates whether y approximates it well: $v_c = 0$ indicates that y approximates the concept fairly well and $v_c = 1$ indicates that it does not. Our goal is to release a differentially private indicator vector v with all the indicator bits for all the functions.

Indicator Vector. To release the indicator bit vector v, we compute for each concept c, the "distance" of the input database D from the output y:

$$d_{c,y}(D) = |c(D) - c(y)|$$

We add binomial noise to these distance counts to ensure privacy (see [5, 8]). The noise is added to ensure that for any collection of at most s_{bad} concepts, their noisy distance counts are $(\varepsilon/2, \exp(-\kappa))$-differentially private. The noise is binomial, generated by adding $4s_{bad} \cdot \kappa^4/\varepsilon^2$ random bits and normalizing by dividing by n. With all but $\exp(-\kappa)$ probability, the noise is thus bounded to be at most:

$$\alpha' = 2\kappa^3 \cdot \sqrt{s_{bad}}/(\varepsilon \cdot n)$$

For a concept $c \in \mathcal{C}$, let u_c be its noisy distance count. While these noisy distance counts are $(\varepsilon/2, \exp(-\kappa))$-private for small (up to s_{bad} size) collections of concepts, the accumulated noisy distance counts for *all* $|\mathcal{C}|$ of the concepts may not be private at all.

Now, for every concept $c \in \mathcal{C}$, we want to release its indicator bit. This is done by rounding the noisy count u_c:

$$v_c = \begin{cases} 0 \text{ if } u_c \leq \alpha + 2\alpha' \\ 1 \text{ if } u_c > \alpha + 2\alpha' \end{cases}$$

The intuition is that if the difference between c's counts on D and y is smaller than $\alpha + 2\alpha'$, then 0 becomes the likely indicator bit, whereas if the difference is larger than $\alpha + 2\alpha'$, then 1 becomes more likely. Now, because the noisy distance counts $\{u_c\}$ are α' accurate for all c's, the indicator bits v_c indeed provide an excellent indication of whether the output y accurately approximates the concept c's count on the database D. In fact, they even an excellent indication of whether y accurately approximates the concept c *on any neighboring database* D'.

CLAIM 4.5. *For any output y, for all concepts c, with all but $\exp(-\kappa)$ probability: (i) if the difference between D and y's counts on c is at most $\alpha + 2/n$ then v_c is 1. (ii) if the difference between D and y's counts on c is at least $\alpha + 4\alpha' - 2/n$ then v_c is 0.*

The above claim shows that for concepts c that y either "fits" very well (about α-close to their counts in D), or for those that y fits very poorly (about $\alpha + 4\alpha'$-far), their indicator bit v_c is with overwhelming probability either 0 or 1. Moreover, this is even true for the given y even when we compute the v_c bits using a neighboring database D' of D! This fact will be useful for analyzing privacy, as essentially the only indicator bits we need to "worry about" when moving from D to D' are the ones whose distance counts (between D and y) are between α and $\alpha + 4\alpha'$. In particular, we only "worry" about concepts for which y is not very accurate w.r.t. D. By the smallness of s_{bad} these concepts are relatively few.

The sanitizer A releases the vector v indicating which concepts y is accurate for. Finally, for each of the concepts c for which $v_c = 1$, it computes c's count in the input D, adds binomial noise a la [5, 8], and releases the noisy count (noise is generated by adding $4s_{bad} \cdot \kappa^4/\varepsilon^2$ random bits).

In summary, A's output includes the synthetic database y, the vector v indicating which concepts y "fits" well, and a collection of noisy counts for all the concepts that y does not fit. Details are omitted for lack of space.

Utility. The probability that there exists a concept c for which y is not $\alpha + 4\alpha'$-accurate but $v_c = 0$ is at most $\exp(-\kappa)$ (Claim 4.5). Thus with overwhelming probability, for the concepts for which $v_c = 0$, the output y indeed gives a $\alpha + 4\alpha'$-accurate count. For the concepts where $v_c = 1$ we output a noisy count using binomial noise as above. This guarantees $(\alpha + \alpha', \exp(-\kappa))$-accuracy for all of these concepts as well. In total, we get $(\alpha + \alpha', \exp(-\kappa))$-utility. Recall that $s_{bad} = |\mathcal{C}| \cdot (\kappa + m)/s_{small}$, and so we get that the sanitizer's accuracy is: $(\alpha + (8\kappa^4 \cdot \sqrt{m}/(\varepsilon \cdot n)) \cdot \sqrt{|\mathcal{C}|/s_{small}}, \exp(-\kappa))$. □

We use binomial noise to sanitize the small concept class C' remaining at the basis of our recursive construction. We call this *"base sanitizer"* A_0. To prove Theorem 4.3 we take $\alpha_0 = 1/\sqrt{n}$, and set $d = \sqrt{\log(|C|)}/2$ and $f = 2^{2\sqrt{\log(|C|)}}$. This yields a sanitizer for concept classes of size $|C|$ whose error $\alpha = \alpha_d$ satisfies:

$$\alpha = O(\frac{\kappa^5 \cdot \log n \cdot \log|C| \cdot \sqrt{\log|X|} \cdot 2^{\sqrt{\log(|C|)}}}{\varepsilon \cdot \sqrt{n}}).$$

4.4 Sanitizing Dense Datasets

We show how to sanitize in time roughly polynomial in $|X|$ (and only logarithmic in $|\mathcal{C}|$) with error roughly $\sqrt{|X|}/n$.

THEOREM 4.6. *Let X be a data universe an $\mathcal{C}$ a concept class. Take κ to be a security parameter. There exists an ε-differentially private sanitizer A with utility:*

$$(\sqrt{|X|} \cdot \log(|X|) \cdot \kappa^2 \cdot \log(|\mathcal{C}|)/(\varepsilon \cdot n), exp(-\kappa))$$

A runs in time $\text{poly}(|X|, n, \log(|\mathcal{C}|), \log(\kappa), \log(1/\varepsilon))$.

The sanitizer A, for each data item $i \in X$, computes how many times i appears in the input database D and adds to this count noise according to the Laplace distribution $Lap(2/\varepsilon)$. The sanitization is the vector u of all these counts.

We can think of a type x (an element) in the universe X as a binary vector, with $x_j = 1$ if $c_j(x) = 1$, and $x_j = 0$ otherwise. For each type x we can define the corresponding *negative type* $-x$ consisting only of 0's and -1's. The sanitizer outputs a collection of counts, one per type $x \in X$. We can create a "pseudo-synthetic" database with the types in $\pm X$.

5. THE TRAITOR TRACING CONNECTION

In this section we show that hardness of sanitizing and traitor tracing are essentially equivalent.

Traitor Tracing. Traitor tracing schemes enable a publisher to trace a pirate decryption box to a secret key (of a treacherous user) that was used to create the box. For example, consider a content provider who wishes to broadcast his content to a group of subscribers. To do this, he gives each of the subscribers a decryption box that uses a secret key. Unfortunately, a subscriber might now build and distribute a pirate decoder. Traitor tracing schemes ensure that if such a pirate decoding box is found, the content distributer can run a *tracing* algorithm to recover at least one private key used to create the box. Tracing schemes were introduced by Chor, Fiat and Naor [4] and there are many constructions with various choices of the parameters (see [2] for recent references).

A (private-key) traitor-tracing scheme consists of algorithms *Setup*, *Encrypt*, *Decrypt* and *Trace*. The *Setup* algorithm generates a key bk for the broadcaster and N subscriber keys $k_1, \ldots, k_N$. The *Encrypt* algorithm encrypts a given bit using the broadcaster's key bk. The *Decrypt* algorithm decrypts a given encryption using any of the subscriber keys. The tracing algorithm gets the key bk and oracle access to a (pirate) decryption box[3] and outputs a key k_i (of some user $i \in \{1, \ldots, N\}$) that was used to create the pirate box.[4] An important parameter of a traitor-tracing scheme is its *collusion-resistance*: a scheme is t-resilient if tracing is guaranteed to work as long as no more than t keys are used to create the pirate decoder. When $t = N$, i.e. tracing works even if all the subscribers join forces to try and create a pirate decoder, the scheme is said to be *fully resilient*. The other parameters that we will be especially interested in are the scheme's ciphertext and private key lengths. A fuller definition follows.

DEFINITION 5.1. *(t-resilient Traitor-Tracing Scheme). A scheme* $(Setup, Encrypt, Decrypt, Trace)$ *is a* t-resilient traitor tracing scheme *if (i) the ciphertexts it generates are semantically secure (see [13]), and (ii) no PPT adversary A can "win" in the following game with non-negligible probability (over the coins of* $Setup, A, Trace$*):*

The adversary A receives the number of users N and a security parameter κ *and (adaptively) requests the keys of up to t users* $\{i_1, \ldots, i_t\}$*. The adversary then outputs a pirate decoder Dec. The Trace algorithm is run with the tracing key and black-box access to Dec, it outputs the key of a user* $i \in [N]$ *error symbol* $\perp$*. We say that an adversary A "wins" if it is both the case that Dec has a non-negligible advantage in decrypting ciphertexts* and *the output of Trace is not in the set of keys of users* $\{i_1, \ldots, i_t\}$*.*

A one-time *t-resilient traitor-tracing scheme is one where semantic security is only guaranteed to hold against adversaries that are given only a single ciphertext (as opposed to being given access to a sequence of encryptions or an encryption oracle).*

5.1 Traitor Tracing Implies Hard Sanitizing

In this section we show, in Theorem 5.1, that traitor tracing schemes can be used to construct hard-to-sanitize distributions. The result is later used, together with the traitor-tracing scheme of [3], to separate efficient and inefficient sanitization.

THEOREM 5.1. *(Traitor Tracing Implies Hard Sanitizing). If there exists a fully resilient (i.e. n-resilient) traitor-tracing scheme with private-key size* $ksize = ksize(n, \kappa)$ *and ciphertext size* $csize = csize(n, \kappa)$*, then there exists a concept class ensemble* $\mathcal{C}$*, where* $\mathcal{C}_n$ *has size* 2^{csize}*, and a data universe ensemble* X*, where* X_n *has size* 2^{ksize}*, and a database distribution ensemble* $\mathcal{D}$*, such that for any noticeable function* $f(n) = 1/poly(\kappa)$ *the ensemble* $\mathcal{D}$ *is* $(\mu = neg(n), \alpha = 1/2 - f(n), \beta = 1 - f(n), \gamma = 0, \mathcal{C})$*-hard-to-sanitize.*

Separating Efficient and Inefficient Sanitization. Using this theorem, together with a traitor-tracing scheme of Boneh, Sahai and Waters [3], we obtain a hardness-of-sanitizing result. BSW construct a fully resilient (for any coalition size) traitor-tracing scheme with private keys of size $O(\kappa)$ and ciphertexts of length $O(\sqrt{n} \cdot \kappa)$, where κ is a security parameter. The security of the scheme relies on several hardness assumptions over bilinear groups of composite order.[5]

Plugging this construction into Theorem 5.1, we obtain for any n and security parameter κ, a data universe X of size $|X| = 2^{\kappa}$, a concept class of size $|\mathcal{C}| = 2^{\sqrt{n} \cdot \kappa}$, and a $(f(n), 1/2 - f(n), 1 - f(n), 0)$-hard-to-sanitize distribution $\mathcal{D}$ over databases of n items from X. Compare this with the (exponential-time) sanitizer of BLR: their sanitizer is guaranteed to give differential privacy as long as the database size satisfies $n = \Theta(\log(|\mathcal{C}|) \cdot \log(|B|))$ which is $\Theta(\sqrt{n} \cdot \kappa^2)$ (here we take $\alpha, \beta, \varepsilon$ to be constant). Taking the security parameter to be a small polynomial in n, we conclude that the task of sanitizing databases drawn from the distribution $\mathcal{D}$ is *information-theoretically possible* (for unbounded sanitizers), but *computationally intractable*:

COROLLARY 5.2. *(Separating efficient and unbounded sanitizers, informal). Under the appropriate hardness assumptions on bilinear groups of composite order, there exists a data universe, a concept class and a database distribution where sanitization is possible in exponential time, but impossible in polynomial time.*

[3]We assume that the box is stateless; this is without loss of generality by the work of Kiayias and Yung [16].

[4]Here we assume the trace algorithm output the key of a user, rather than just the user's name. The two definitions are equivalent as all keys can be included in the tracing key.

[5]Specifically, they need to assume the Decision 3-Party Diffie-Hellman Assumption, the Subgroup Decision Assumption, and the Bilinear Subgroup Decision Assumption. See [3] for a precise statement of these assumptions and the results.

PROOF IDEA FOR THEOREM 5.1. Take a traitor tracing scheme $(Setup, Encrypt, Decrypt, Trace)$ as above. We construct a hard to sanitize concept class and database distribution.

The data universe. Consider the data universe of possibly keys $X = \{0,1\}^{ksize(n+1,\kappa)}$.

The concept class. $\mathcal{C}$ will contain a concept for every possible ciphertext. I.e. for every $m \in \{0,1\}^{csize(n+1,\kappa)}$. The concept c_m on input a key-string k outputs the decryption of m using the key k (if the messages are longer than one bit, output the least significant bit).

Hard-to-sanitize distribution. The distribution $\mathcal{D}$ on databases uses *Setup* to generate $n+1$ decryption keys for the users, n of these keys are used (at random) to generate the n entries of the database D. The $n+1$-th key is the extra item D'_i.

Intuition. The intuition is that for a concept corresponding to a valid encryption of a 0 or 1 message, all the keys in the database decrypt it correctly and output 0 or 1 respectively. So we can view any sanitizer that maintains $(1/2 - f(n), f(n), 0)$ utility as an adversary that with probability $f(n)$ outputs an object that decrypts encryptions of 0 or 1 correctly. We can use the traitor-tracing *Trace* algorithm on such a sanitizer to trace one of the keys in the input of the sanitizer. More formally, we prove the two claims below:

CLAIM 5.3. *For any alleged PPT sanitizer A with $(\alpha = 1/2 - f(n), \beta = f(n), \gamma = 0)$-utility, where $f(n)$ is noticeable, the probability (over $(D, D'_i) \sim \mathcal{D}$ and A's coins) that (A maintains α-utility) and ($D \cap Trace(A(D)) = \emptyset$) is at most $neg(\kappa)$.*

CLAIM 5.4. *For any PPT A and any $i \in [n]$:*

$$\Pr_{(D,D'_i)\sim\mathcal{D},\ E\text{'s coins}}\left[D_i \in Trace(A(D'))\right] = neg(\kappa)$$

We conclude that the database distribution $\mathcal{D}$ is $(neg(\kappa), 1/2 - f(n), f(n), 0, \mathcal{C})$-hard-to-sanitize. □

5.2 Hard Sanitizing Implies Traitor Tracing

In this section we show that hard-to-sanitize database distributions can be used to construct traitor-tracing schemes. Note that we require that it is hard to sanitize even with a small (but noticeable) γ, i.e. it is hard to sanitize while maintaining utility for all but a small fraction of the concepts. It is interesting to ask whether the same holds even for smaller γ or even $\gamma = 0$.

THEOREM 5.5. *(Hard Sanitizing Implies Traitor Tracing). Let $\mathcal{C}$ be an ensemble of concept classes, X an ensemble of data universes, and $\mathcal{D}$ a distribution that for every noticeable function $f(n) = 1/\text{poly}(n)$ is $(\mu = neg(n), \alpha = O(1/\log(n)), \beta = 1 - f(n), \gamma = O(1/\log(n)), \mathcal{C})$-hard-to-sanitize. Then there exists an $n/\text{polylog}(n)$-resilient one-time traitor tracing scheme, with private keys of size $\log(|X|)\cdot\text{polylog}(n)$ and cipher-text length $\log(|\mathcal{C}|)\cdot\text{polylog}(n)$. The scheme is secure against time $\text{poly}(n, \log(|\mathcal{C}|), \log(|X|))$ adversaries.*

PROOF. (Intuition). For a hard-to-sanitize distribution, we are guaranteed that for any sanitizer with good utility we can "trace" one of the items in its input. Given a hard-to-sanitize database distribution and concept class, the idea will be to use a randomly drawn n-item database as a "master key", where the secret used to decrypt messages is the counts of random concepts on this database. To give users private keys, we can randomly partition the database into $n/\text{polylog}(n)$ sets of $\text{polylog}(n)$ items each, each such set will be a "key". These sets are large enough that with overwhelming probability their counts on a random collection of say $\text{polylog}(n)$ concepts are *all* close to the counts of the original database. We will to design an encryption scheme where decryption is equivalent to computing approximate counts on random concepts. Once we do this, the intuition is that a decryption box can be used to compute approximate counts, viewing this box as a sanitization of the database we conclude (because sanitizing is hard) that the decryption box can be "traced" to the keys (database items) that were used to create it.

In order to ensure that all keys decrypt the same way we *add noise and round* the counts in a way that ensures that as long as two sets of counts are close they will (with very high probability) be rounded to *exactly* the same rounded counts (in a different setting, a similar idea was used by Saks and Zhou [18]). Now we can extract a hardcore bit of the noisy rounded counts and use it to encrypt a message, and all of the users will be able to decrypt a ciphertext (except with some low error probability).

Using the above scheme, any decryption box can be used to compute approximate counts for a large random collection of concepts with noticeable probability. We use a uniform direct-product theorem of [14] to show that this, in turn, means that a decryption box can be used to build a procedure that with noticeable probability computes approximate counts correctly for all but a small fraction of the concepts. We can treat such a procedure as an alleged sanitizer, and from the hardness of sanitizing the database distribution we can use the algorithm T (from the hardness of sanitizing guarantee) to "trace" a database item in the procedure's input. The user whose key contains that item must have colluded to build the decryption box. A full description and proof follow.

The Scheme. We now describe the *Setup*, *Encrypt* and *Decrypt* algorithms of the traitor tracing scheme. We first analyze these algorithms and then present the *Trace* algorithm and complete the proof of the theorem. Throughout this section "efficient" algorithms are those that run in time $\text{poly}(n, \log(|\mathcal{C}|), \log(|X|))$.

The *Setup* Algorithm Choose an n-item database D from the distribution $\mathcal{D}$. Take $s = \text{polylog}(n)$ to be a size for subsets of items from D. The algorithm forms n/s keys by dividing at random the n items in D into n/s disjoint subsets, each of s items. The i-th player's key is just the i-th subset, and so keys are indeed of size $\log(|X|)\cdot polylog(n)$. The broadcast key is simply the collection of all secret keys.

The *Encrypt* Algorithm receives a message bit b to encrypt and a private key which is just a set of s items.[6] It chooses uniformly and at random $\ell = \text{polylog}(n)$ concepts $c_1, \ldots, c_\ell \in \mathcal{C}$. For each of these concepts, it com-

[6]In traitor tracing schemes encryption is usually done using a broadcast key. We construct a scheme where any private key can be used for encryption.

putes the fraction of items in the key that satisfy this concept, and then chooses a uniformly random noise value $t \in \{-50/\log(n), -49/\log(n), \ldots, 50/\log(n)\}$ and adds t to all the fractional counts. The *Encrypt* algorithm then rounds all the (noisy) fractional counts down to the nearest multiple of $100/\log(n)$. Let the resulting vector of rounded noisy fractional counts be $v = (v_1, \ldots, v_\ell)$.

The *Encrypt* algorithm extracts a hard-core bit of v and XORs this hard-core bit with the message bit b in order to encrypt it. To do this, it chooses a random bit vector r s.t. $|r| = |v|$ and outputs the encryption:

$$Enc_{key}(b) = (c_1, \ldots, c_\ell, t, r, \langle v, r\rangle \oplus b)$$

The *Decrypt* **Algorithm** gets as input a private key and a ciphertext $(c_1, \ldots, c_\ell, t, r, cipher)$. It proceeds similarly to the *Encrypt* algorithm and computes, using the items from its given private key and $(c_1, \ldots, c_\ell, t)$ from the ciphertext, a vector of rounded noisy fractional counts v'. It then outputs the decrypted message as: $cipher \oplus \langle v', r\rangle$.

CLAIM 5.6. *For any message b, with probability 9/10 over Setup's and Encrypt's coins, for any pair of private keys k_i, k_j we get that $Decrypt(k_j, Encrypt(k_i, b)) = b$.*

We now present a helpful claim that will be used to argue both semantic security and traitor tracing (with good parameters). We will need to transform an algorithm that computes correct approximate counts simultaneously on $\ell = \text{polylog}(n)$ randomly chosen concepts $(c_1, \ldots, c_\ell)$ with non-negligible probability, into a procedure that with noticeable probability over some initial coin tosses correctly computes approximate counts on all but a small fraction of the concepts. This is essentially a *hardness amplification* question, and we use the recent results of [14], which we restate here for the case of amplifying the hardness of computing approximate counts (usually hardness amplification is used to amplify the hardness of *precisely* computing a function).

CLAIM 5.7 ([14] RESTATED). *Let M be an efficient procedure that with noticeable probability $1/q(n)$, given $\ell = \text{polylog}(n)$ random concepts $(c_1, \ldots, c_l)$, achieves α-utility on all of the concepts. Then M can be used to construct an efficient procedure M' such that with probability $\Omega(1/q(n))$ (over some initial coin tosses), the procedure M' achieves $3 \cdot \alpha$-utility for all but a $1/\log(n)$-fraction of the concepts.*

Tracing. We are now ready to present the *Trace* algorithm. Note that semantic security follows from the tracing property. The idea is to transform an adversary A that creates a decryption box into an alleged sanitizer for the database D used to generate the keys. We can then use the algorithm T from the hardness-of-sanitization guarantee to trace one of the database items that was in A's collection of keys. Throughout the analysis we assume that for the D used to generate the keys (i.e. chosen according to the hard-to-sanitize distribution $\mathcal{D}$), indeed the two conditions of hardness-of-sanitizing (Conditions 1 and 2 of Definition 2.3) hold. This is guaranteed to occur except with negligible probability.

Let A be an adversary that asks for some of the decryption keys and creates (with noticeable probability $1/p(n)$) a decryption box M that decrypts a random ciphertext correctly with probability $1/2 + \lambda$.

A decryption box can be used to predict the counts of a collection of ℓ concepts with noticeable probability. The *Trace* algorithm first uses the box M to generate a polynomial size list of algorithms $\{M_1, \ldots, M_{\text{poly}}\}$, one of which correctly predicts the count of a random concept with very high probability. We treat each of these algorithms as a sanitization. *Trace* runs T on each of the algorithms M_j and outputs the key (say the first) that contains an item that was in T's output when run on one of these algorithms (or $\perp$ if such a user does not exist).

The analysis will use the fact that one of the M_j's has good utility, and so from the hardness of sanitizing, with overwhelming probability T's output on the "good" M_j has an intersection with one of the keys. We will also show that with high probability all the items in T's output that are also in D are ones that were in keys used by the traitor-tracing adversary A.

CLAIM 5.8. *Let A be any efficient traitor-tracing adversary that with noticeable probability $1/p(n)$ outputs a decryption box M that has noticeable decryption advantage λ.*

With all but negligible probability, when A generates a decryption box M with decryption advantage λ, the Trace procedure run on M outputs one of the keys used by A.

PROOF. We use the algorithm T as a traitor-tracing procedure on the decryption box M. As a first step, we view A (the traitor-tracing adversary) as an alleged sanitizer. We know that with probability $1/p(n)$ over the the selection of D and the adversary A's coins, the adversary outputs a "good" decryption box M. This means that for a random tuple $(c_1, \ldots, c_\ell, t)$, with probability at least $\lambda/2$ over the tuple, the distinguishing advantage of M on random ciphertexts created using that tuple is at least $\lambda/2$. By the Goldreich-Levin Theorem, we can use M to output a set of $O(1/\lambda^2)$ count vectors. With high probability, one of these count vectors will be the correct rounded noisy fractional count vector and is thus a $100/\log(n)$-approximation to the counts of the database D on *all of the concepts* $(c_1, \ldots, c_\ell)$.

The *Trace* algorithm can now use the box M to generate a list of $s = O(n/\lambda^2)$ boxes $M_1, \ldots M_s$ such that with overwhelming probability one of the boxes, say M_j, predicts the count of a random concept with good accuracy with very high probability. We treat each of these boxes as a sanitization, and we treat A as a sanitizer that outputs one of the boxes at random. We can thus run the algorithm T on all of these alleged sanitizations.

With probability $1/(s \cdot p(n))$ over the selection of D and the coins of A, it outputs a "good" box M_j, which is a sanitization that is $O(1/\log(n))$-accurate for all but a $O(1/\log(n))$ fraction of the concepts. Viewing A as a sanitizer, by the $(neg(n), \alpha = O(1/\log(n)), \beta = 1 - f(n), \gamma = O(1/\log(n)))$-hardness of sanitizing $\mathcal{D}$, and as long as $f(n) \geq p(n) \cdot n/\lambda^2$, we get that this alleged sanitizer falls within the hard-to-sanitize regime of parameters. This means that when M is a decryption box with advantage λ, when we run T on the "good" box M_j we constructed from M, with overwhelming probability it outputs some element $D_k \in D$. In particular, with overwhelming probability *Trace* will output the name of *some* user (who has item D_k in his key). It remains to show that with high probability this user was indeed a colluder, i.e. that all of the items in T's output that are also in D are ones that were in user keys used by the traitor-tracing adversary A.

First note that if A got *all* the keys as input then we are done. It remains to show that when A only asks for some (but not all) of the keys, T only outputs items of D that are in one of A's input keys (except with negligible probability).

Suppose this is not the case: A (adaptively) chooses a set S of not all the database items, and outputs a decryption box M with advantage λ. When we run *Trace* on this box, with noticeable probability it outputs an item $D_k \in D$ such that $D_k \notin S$. This will contradict Condition 2 of the hardness of sanitizing $\mathcal{D}$. Let us re-examine that condition, it stipulates that if we replace any $D_i \in D$ with the (unique) item D'_i, the probability that, when we run T on the output of *any efficient procedure* on input D', the output contains D_i is negligible.

To see that A as above contradicts condition 2, consider running the *Trace* algorithm to generate the list of boxes $M_1, \ldots, M_s$ and running T on one of these boxes at random. We assumed (for contradiction), that with noticeable probability T's output contains D_k that was not in the input item set S. Now choose at random an item $i \in [n]$ and replace D_i with D'_i to obtain a new database D'. Whenever the above event occurred and T's output contained D_k, if we run the same procedure on the *modified* D' (where we choose i at random and replace D_i), the probability that $i = k$ and the procedure's output still contains D_i is at least $1/n$ (because i was chosen uniformly at random). In total, the probability that the output of T on a random box from the collection $M_1, \ldots, M_s$ contains D_i is noticeable, contradicting Condition 2 of the hardness of sanitizing $\mathcal{D}$. □

6. CONCLUSIONS AND OPEN PROBLEMS

We have provided an almost complete characterization of when general methods for efficiently manufacturing a synthetic database maintaining approximately correct fractional counts for *all* concepts in $\mathcal{C}$, while ensuring a strong privacy guarantee, exist. We have shown an equivalence between hardness of sanitizing with an arbitrary data structure and the existence of traitor tracing schemes.

Is it possible to lower the dependence of the error in our positive results to polylogarithmic in the sizes of the universes of concepts and data? It is conceivable that a different type of recursive algorithm, perhaps with fewer calls to the linear programming procedure for synthesizing database (Section 4.1) or a better method for creating the indicator vector v, would yield such a result. It is also worth exploring the connection to boosting algorithms [19].

Are there tracing traitor schemes that are *fully* resilient and yet have keys and ciphertext lengths independent of the number of users? This would yield a relatively small concept class ensemble that cannot be sanitized. Another implication for such schemes would be for showing hardness of learning privately, an issue considered by [15].

Finally, there are many domains where we do not know of an *efficient* privacy preserving sanitizer. One example is that of half-spaces, where the concepts are half-spaces and the data items are points. [1] gave an efficient algorithm, but one that answers not necessarily for the given half-space query but for a near half space. There are many other geometric type queries where we do not know whether a good algorithm exists and it is conceivable that the tools and mechanism developed for our main positive result would be helpful, especially since they are not inherently expensive (e.g. sampling and rounding). See [10] for very recent work on differentially private *coresets*; these are weighted pointsets, essentially synthetic databases for geometric problems.

7. REFERENCES

[1] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[2] D. Boneh and M. Naor. Traitor tracing with constant size ciphertext. In *ACM Conference on Computer and Communications Security*, pages 501–510, 2008.

[3] D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, pages 573–592, 2006.

[4] B. Chor, A. Fiat, and M. Naor. Tracing traitors. In *CRYPTO*, pages 257–270, 1994.

[5] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevy and T. Rabin, editors, *First Theory of Cryptography Conference (TCC)*, volume 3876, pages 265–284. Springer-Verlag, 2006.

[7] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of lp decoding. In *STOC*, pages 85–94, 2007.

[8] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, pages 528–544, 2004.

[9] C. Dwork and S. Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *CRYPTO*, pages 469–480, 2008.

[10] D. Feldman, A. Fiat, H. Kaplan, and K. Nissim. Private coresets. These Proceedings, 2009.

[11] O. Goldreich. *The Foundations of Cryptography - Volume 2*. Cambridge University Press, 2004.

[12] O. Goldreich, S. Goldwasser, and S. Micali. How to construct pseudorandom functions. *Journal of the ACM*, 33(2):792–807, 1986.

[13] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[14] R. Impagliazzo, R. Jaiswal, V. Kabanets, and A. Wigderson. Uniform direct product theorems: simplified, optimized, and derandomized. In *STOC*, pages 579–588, 2008.

[15] P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *FOCS*, pages 1–19, 2008.

[16] A. Kiayias and M. Yung. Self protecting pirates and black-box traitor tracing. In *CRYPTO*, pages 63–79, 2001.

[17] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE Computer Society, 2007.

[18] M. E. Saks and S. Zhou. Bp $_{\mathrm{h}}$space(s) subseteq dspace($s^{3/2}$). *J. Comput. Syst. Sci.*, 58(2):376–403, 1999.

[19] R. E. Schapire. Theoretical views of boosting and applications. In *ATL*, pages 13–25, 1999.

[20] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.