Answering n^{2+o(1)} Counting Queries with Differential Privacy is Hard[*]

Jonathan Ullman[†]
Harvard University
jullman@seas.harvard.edu

ABSTRACT

A central problem in differentially private data analysis is how to design efficient algorithms capable of answering large numbers of *counting queries* on a sensitive database. Counting queries are of the form "What fraction of individual records in the database satisfy the property q?" We prove that if one-way functions exist, then there is no algorithm that takes as input a database $D \in (\{0,1\}^d)^n$, and $k = \widetilde{\Theta}(n^2)$ arbitrary efficiently computable counting queries, runs in time $\mathrm{poly}(d, n)$, and returns an approximate answer to each query, while satisfying differential privacy. We also consider the complexity of answering "simple" counting queries, and make some progress in this direction by showing that the above result holds even when we require that the queries are computable by constant-depth (AC^0) circuits.

Our result is almost tight because it is known that $\widetilde{\Omega}(n^2)$ counting queries can be answered efficiently while satisfying differential privacy. Moreover, many more than n^2 queries (even exponential in n) can be answered in exponential time.

We prove our results by extending the connection between differentially private query release and cryptographic traitor-tracing schemes to the setting where the queries are given to the sanitizer as input, and by constructing a traitor-tracing scheme that is secure in this setting.

Categories and Subject Descriptors

F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

Keywords

Differential Privacy; Traitor-Tracing

1. INTRODUCTION

Consider a database $D \in (\{0,1\}^d)^n$, in which each of the n 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. It is especially desirable to achieve *differential privacy* [7], which guarantees that no individual's data has a significant influence on the information released about the database.

Some of the most basic statistics on a database are *counting queries*, which are queries of the form, "What fraction of individual records in D satisfy some property q?" In particular we would like to construct differentially private **sanitizers** that, given a database D and k counting queries $q_1, \ldots, q_k$ from a family $\mathcal{Q}$, outputs an approximate answer to each of the queries. We would like the number of queries, k, to be as large as possible, and the set of feasible queries, $\mathcal{Q}$, to be as general as possible. Ideally, $\mathcal{Q}$, would contain all counting queries.[1] Moreover, we would like the algorithm to run as efficiently as possible.

Some of the earliest work in differential privacy [7] gave an efficient sanitizer—the so-called *Laplace Mechanism.* The Laplace Mechanism answers any set of k arbitrary efficiently computable counting queries by perturbing the answers with appropriately calibrated random noise, providing good accuracy (say, within $\pm.01$ of the true answer) as long as $k \lesssim n^2$.

The ability to approximately answer n^2 counting queries is quite powerful, especially in settings where data is abundant and n is large. However, being limited to n^2 queries can be restrictive in settings where data is expensive or otherwise difficult to acquire, and n is small. It can also be restrictive when the budget of queries is shared between multiple analysts. Fortunately, a remarkable result of Blum et al. [2] (with subsequent developments in [8, 10, 17, 14, 12, 13]), showed that differentially private algorithms are not limited to n^2 queries. They showed how to approximately answer arbitrary counting queries even when k is *exponentially larger* than n. Unfortunately, their algorithm, and all subsequent algorithms capable of answering more than n^2 arbitrary counting queries, run in time (at least) $\mathrm{poly}(2^d, n, k)$.

The result of Blum et al., raises the exciting possibility of an *efficient* algorithm that can privately compute approximate answers to large numbers of counting queries. Unfortu-

[*]A full version of this work appears on arXiv [20]

[†]Supported by NSF grants CNS-0831289, CNS-1237235, and a gift from Google, Inc.

[1]It may require super-polynomial time just to evaluate an arbitrary counting query, which would rule out efficiency for reasons that have nothing to do with privacy. For this discussion, we restrict attention to queries that are efficiently computable, so are not the bottleneck in the computation.

nately, Dwork et al. [8] gave evidence that efficient sanitizers are inherently less powerful than their computationally unbounded counterparts. They study the problem of construcing differentially private **one-shot sanitizers** that, given a database D, produce a summary from which approximate answers to *every* query in $\mathcal{Q}$ can be computed, while both the sanitizer and the summary run in time much less than the size of $\mathcal{Q}$. Dwork et al. constructed a family of $2^{\widetilde{O}(\sqrt{n})}$ queries for which there is no efficient (time $\mathrm{poly}(d,n)$) one-shot sanitizer (under certain cryptographic assumptions), even though there is an inefficient (time $\mathrm{poly}(2^d, n, |\mathcal{Q}|)$) one-shot sanitizer even if $|\mathcal{Q}|$ is nearly 2^n. For any family $\mathcal{Q}$, constructing an efficient one-shot sanitizer is one way of constructing an efficient sanitizer that answers any polynomial number of queries from $\mathcal{Q}$. Thus, hardness results for one-shot sanitizers rule out a particular way of constructing efficient sanitizers. However, ultimately a polynomial-time analyst will only be able to ask a polynomial number of queries, and hardness results for one-shot sanitzers still leave hope that there might be an efficient sanitizer that can answer many more arbitrary counting queries than the Laplace Mechanism.

Unfortunately, we show that this is not the case—there is no efficient, differentially private algorithm that takes a database $D \in (\{0,1\}^d)^n$, and $\widetilde{\Theta}(n^2)$ arbitrary, efficiently computable counting queries as input and outputs an approximate answer to each of the queries. One way to summarize our results is that, unless we restrict the set $\mathcal{Q}$ of allowable queries, or allow exponential running time, then the Laplace Mechanism is essentially the best possible algorithm for answering counting queries with differential privacy.

1.1 Our Results and Techniques

As discussed above, in this paper we give new hardness results for answering counting queries while satisfying differential privacy. To make the statement of our results more concrete, we will assume that the counting queries are given to the sanitizer as input in the form of circuits that, on input an individual record $x \in \{0,1\}^d$, decide whether or not the record x satisfies the property q. We say the queries are efficiently computable if the corresponding circuits are of size $\mathrm{poly}(d,n)$.

Theorem 1.1. *Assuming the existence of one-way functions, there is no algorithm that, on input a database $D \in (\{0,1\}^d)^n$ and $\widetilde{\Theta}(n^2)$ efficiently computable counting queries, runs in time $\mathrm{poly}(d,n)$ and returns an approximate answer to each query to within $\pm.49$, while satisfying differential privacy.*

In particular, Theorem 1.1 applies to **interactive sanitizers**, which are sanitizers that receive (possibly adaptively chosen) queries one at a time. Many positive results achieve this stronger notion of sanitization. In particular, the Laplace mechanism is an efficient interactive sanitizer that answers $\widetilde{\Omega}(n^2)$ queries and there exist interactive sanitizers that can answer nearly 2^n queries in time $\mathrm{poly}(2^d, n)$ per query interactively [17, 14, 12].

We also show that, the same theorem holds even for queries that are computable by unbounded-fan-in circuits of depth 6 over the basis $\{\wedge, \vee, \neg\}$ (a subset of the well-studied class AC^0), albeit under a stronger (but still plausible) cryptographic assumption.

Theorem 1.2. *Under the assumptions described in Section 5.6, there is no algorithm that, on input a database $D \in (\{0,1\}^d)^n$ and $\widetilde{\Theta}(n^2)$ efficiently computable depth-6 queries (circuits), runs in time $\mathrm{poly}(d,n)$ and returns an approximate answer to each query to within $\pm.49$, while satisfying differential privacy.*

Theorem 1.2 should be contrasted with the results of Hardt, Rothblum, and Servedio [15] as well as Thaler, Ullman, and Vadhan [19], which give efficient sanitizers for answering $n^{\Omega(\sqrt{k})} \gg n^2$ monotone k-way conjunction queries, a much simpler class than polynomial-size depth-6 circuits.[2]

We now describe our techniques.

The Connection with Traitor-Tracing.

We prove our results by building on the connection between differentially private sanitizers for counting queries and *traitor-tracing schemes* discovered by Dwork et al. [8]. Traitor-tracing schemes were introduced by Chor, Fiat, and Naor [5] for the purpose of identifying pirates who violate copyright restrictions. Roughly speaking, a (fully collusion-resilient) traitor-tracing scheme allows a sender to generate keys for n users so that 1) the sender can broadcast encrypted messages that can be decrypted by any user, and 2) any *efficient pirate decoder* capable of decrypting messages can be *traced* to at least one of the users who contributed a key to it, even if an arbitrary coalition of the users combined their keys in an arbitrary efficient manner to construct the decoder.

Dwork et al. show that the existence of traitor-tracing schemes implies hardness results for one-shot sanitizers. Very informally, they argue as follows: Suppose a coalition of users takes their keys and builds a database $D \in (\{0,1\}^d)^n$ where each record contains one of their user keys. The family $\mathcal{Q}$ will contain a query q_c for each possible ciphertext c. The query q_c asks "What fraction of the records (user keys) in D would decrypt the ciphertext c to the message 1?" Every user can decrypt, so if the sender encrypts a message $m \in \{0,1\}$ as a ciphertext c, then every user will decrypt c to m. Thus the answer to the counting query, q_c, will be m.

Suppose there were an efficient one-shot sanitizer for $\mathcal{Q}$. Then the coalition could use it to efficiently produce a summary of the database D that enables one to efficiently compute an approximate answer to every query q_c, which would also allow one to efficiently decrypt the ciphertext. Such a summary can be viewed as an efficient pirate decoder, and thus the tracing algorithm can use the summary to trace one of the users in the coalition. However, if there is a way to identify one of the users in the database from the summary, then the summary is not differentially private.

In order to instantiate their result, they need a traitor-tracing scheme. Since $\mathcal{Q}$ contains a query for every ciphertext, the parameter to optimize is the length of the ciphertexts. Using the fully collusion-resilient traitor-tracing scheme of Boneh, Sahai, and Waters [3], which has ciphertexts of length $\widetilde{O}(\sqrt{n})$, they obtain a family of queries of size $2^{\widetilde{O}(\sqrt{n})}$ for which there is no efficient one-shot sanitizer. Dwork et al. also discovered a converse—proving hardness

[2]A monotone k-way conjunction query on a database $D \in (\{0,1\}^d)^*$ is specified by a set of positions $S \subseteq [d]$, $|S| = k \leq d$, and asks "What fraction of records in D have a 1 in every position in S?".

of one-shot sanitization for a smaller family of queries requires constructing traitor-tracing schemes with shorter ciphertexts, which is a seemingly difficult open problem.

Our Approach.

In our setting of sanitization (rather than one-shot sanitization), we don't expect to answer every query in $\mathcal{Q}$, only a much smaller set of queries requested by the analyst. At first glance, this should make answering the queries much easier, and thus make it more difficult to demonstrate hardness. However, the attacker does have the power to choose the queries to which he wants answers, and can choose queries that are most difficult to sanitize. Our first observation is that in the traitor-tracing scenario, the tracing algorithms only query the pirate decoder on a polynomial number of ciphertexts, which are randomly chosen and depend on the particular keys that were instantiated for the scheme. For many schemes, even $\widetilde{O}(n^2)$ queries is sufficient. Thus it would seem that the tracing algorithm could simply decide which queries it will make, give those queries as input to the sanitizer, and then use the answers to those queries to identify a user and violate differential privacy.

However, this intuition ignores an important issue. Many traitor-tracing schemes (including [3]) can only trace *stateless* pirate decoders, which essentially commit to a response to each possible query (or a distribution over responses) once and for all. For one-shot sanitizers, the private summary is necessarily stateless, and thus the result of Dwork et al. can be instantiated with any scheme that allows tracing of stateless pirate decoders. However, an arbitrary sanitizer might give answers that depend on the sequence of queries. Thus, in order to prove our results, we will need a traitor-tracing scheme that can trace *stateful* pirate decoders.

The problem of tracing stateful pirates is quite natural even without the implications for private data analysis. Indeed, this problem has been studied in the literature, originally by Kiayias and Yung [16]. They considered pirates that can *abort* and *record history*. However, their solution, and all others known, does not apply to our specific setting due to a certain "watermarking assumption" that doesn't apply when proving hardness-of-sanitization (see discussion below). To address this problem, we also refine the basic connection between traitor-tracing schemes and differential privacy by showing that, in many respects, fairly weak traitor-tracing schemes suffice to establish the hardness of preserving privacy. In particular, although the pirate decoder obtained from a sanitizer may be stateful and record history, the accuracy requirement of the sanitizer means that the corresponding pirate decoder cannot abort, which will make it easier to construct a traitor-tracing scheme for these kinds of pirates. Indeed, we construct such a scheme to establish Theorem 1.1.

The scheme also has weakened requirements in other respects, having nothing to do with the statefulness of the pirate or the tracing algorithm. These weakened requirements allow us to reduce the complexity of the decryption, which means that the queries used by the attacker do not need to be arbitrary polynomial-size circuits, but instead can be circuits of constant depth, which allows us to establish Theorem 1.2. Another technical issue arises in that all k queries must be given to the sanitizer at once, whereas tracing algorithms typically are allowed to query the pirate interactively. However, we are able to show that the scheme we construct can be traced using one round of queries. See Sections 3.1 and 4 for a precise statement of the kind of traitor-tracing scheme that suffices and Section 5 for our construction.

Our construction is based on a well-known fully collusion resilient traitor-tracing scheme [5], but with a modified tracing algorithm. The tracing algorithm uses *fingerprinting codes*, which have been employed before in the context of traitor-tracing and content distribution, but our tracing algorithm is different from all those we are aware of. It is not so surprising that the scheme doesn't appear in the literature, since it is only traceable against a weak form of stateful adversary, and doesn't achieve novel parameters or functionality. The motivation for constructing a new scheme is to allow for tracing with a minimal number of non-adaptively chosen queries, to achieve tracing without context-specific watermarking assumptions, and to simplify the decryption circuit (at the expense of weakening the security parameters and functionality). None of these features are especially desirable in the setting of content distribution, which explains why the scheme was not previously known.

1.2 Additional Related Work

Stronger hardness results are known for sanitizers whose output is a "synthetic database"—roughly, a new database (of the same dimensions) that approximately preserves the answer to some set of queries. Ullman and Vadhan [21], building on Dwork et al. [8], showed that it is hard to generate a private synthetic database that is accurate for essentially any non-trivial family of queries! This barrier applies even to families of queries of size $\ll n^2$, for which efficient sanitizers that do not output synthetic data are known (e.g. the Laplace mechanism). Thus these results say more about the hardness of synthetic data and the limitations of current techniques than they do about the hardness of answering large numbers of counting queries.

Gupta et al. [11] considered algorithms that access the database only by making a polynomial number of "statistical queries" (essentially counting queries). They showed that such algorithms cannot be a one-shot sanitizer (even a non-privacy-preserving one!) that approximately answers certain simple families of counting queries with high accuracy. This barrier only applies to algorithms that run in time sublinear in the number of queries, which is not something we expect in our setting.

Dwork, Naor, and Vadhan [9] gave information theoretic lower bounds for *stateless sanitizers*, which take k queries as input, but whose answers to each query do not depend on the other $k-1$ input queries. They showed that (even computationally unbounded) stateless sanitizers can answer at most $\widetilde{O}(n^2)$ queries with non-trivial accuracy, while satisfying differential privacy. Although their result is information theoretic, and considers a highly restricted type of sanitizer, their techniques are related to ours. We elaborate on this connection in the full version of this work.

As we mentioned earlier, there has also been considerable interest, and some success, in constructing efficient sanitizers for very simple classes of counting queries such as conjunctions [11, 15, 19], decision lists [19], and halfspaces [2]. There is a significant gap between these results—which apply to families of size $2^{\mathrm{poly}(d)}$ or even just $\mathrm{poly}(d)$—and our lower bound for constant-depth circuits of size $\mathrm{poly}(d, n)$, which would be very interesting to try and close.

2. PRELIMINARIES

Differentially Private Algorithms.

Let a *database* $D \in (\{0,1\}^d)^n$ be a collection of n rows $(x^{(1)}, \ldots, x^{(n)}) \in \{0,1\}^d$. We say that two databases $D, D' \in (\{0,1\}^d)^n$ are *adjacent* if they differ only on a single row, and we denote this by $D \sim D'$.

Definition 2.1 ([7]). *Let $\mathcal{M}\colon (\{0,1\}^d)^n \to \mathcal{R}$ be a randomized algorithm that takes a database as input (where n and d are varying parameters). $\mathcal{M}$ is (ε, δ)-*differentially private *if for every two adjacent databases $D \sim D'$ and every subset $S \subseteq \mathcal{R}$,*

$$\Pr[\mathcal{M}(D) \in S] \le e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta.$$

If $\mathcal{M}$ is (ε, δ)-differentially private for some functions $\varepsilon = \varepsilon(n) = O(1)$, $\delta = \delta(n) = o(1/n)$, we will drop the parameters ε and δ and say that $\mathcal{M}$ is differentially private.

The parameters $\varepsilon = O(1), \delta = o(1/n)$ are essentially the weakest possible, as (ε, δ)-differentially privacy is not a satisfactory privacy guarantee for $\varepsilon = \omega(1)$ or $\delta = \Omega(1/n)$. That our lower bounds apply to the parameters specified in Definition 2.1 makes our results stronger. Most positive results achieve stronger privacy parameters.

Sanitizers for Counting Queries.

Since an algorithm that always outputs $\perp$ satisfies Definition 2.1, we also need to specify what it means for the sanitizer to be useful. In this paper we study sanitizers that give accurate answers to *counting queries.* A counting query on $\{0,1\}^d$ is defined by a predicate $q\colon \{0,1\}^d \to \{0,1\}$. Abusing notation, we define the evaluation of the query q on a database $D = (x^{(1)}, \ldots, x^{(n)}) \in (\{0,1\}^d)^n$ to be $q(D) = \frac{1}{n}\sum_{i=1}^{n} q(x^{(i)})$ We will use $\mathcal{Q}^{(d)}$ to denote a set of counting queries on $\{0,1\}^d$ and $\mathcal{Q} = \bigcup_{d \in \mathbb{N}} \mathcal{Q}^{(d)}$.

We are interested in *sanitizers* that take a sequence of queries from some set $\mathcal{Q}$ as input. Formally a sanitizer is a function $\mathcal{M}\colon (\{0,1\}^d)^n \times (\mathcal{Q}^{(d)})^k \to \mathbb{R}^k$ (where, again, n, d, and k are varying parameters). Notice that we assume that $\mathcal{M}$ outputs k real-valued answers. Think of the j-th component of the output of $\mathcal{M}$ as an answer to the j-th query. For the results in this paper, this assumption will be without loss of generality.[3] Definition 2.1 extends naturally to sanitizers by requiring that for every $q_1, \ldots, q_k \in \mathcal{Q}$, the sanitizer $\mathcal{M}_{q_1,\ldots,q_k}(\cdot) = \mathcal{M}(\cdot, q_1, \ldots, q_k)$ is (ε, δ)-differentially private as a function of the input database.

Now we formally define what it means for a generic sanitizer to give accurate answers.

Definition 2.2 (Accuracy). *Let D be a database and $q_1, \ldots, q_k$ be a set of counting queries. A sequence of answers $a_1, \ldots, a_k$ is* α-accurate *for $q_1, \ldots, q_k$ on D if*

$$\forall j \in [k], |q_j(D) - a_j| \le \alpha.$$

[3]In certain settings, $\mathcal{M}(D, q_1, \ldots, q_k)$ is allowed to output a "summary" $z \in \mathcal{R}$ for some range $\mathcal{R}$. In this case, we would also require that there exists an "evaluator" $\mathcal{E}\colon \mathcal{R} \times \mathcal{Q} \to \mathbb{R}$ that takes a summary and a query and returns an answer $\mathcal{E}(z, q) = a$ that approximates $q(D)$. The extra generality is used to allow $\mathcal{M}$ to run in less time than the number of queries it is answering (e.g. releasing a fixed family of queries), but this is not relevant for our range of parameters where $k = \widetilde{O}(n^2)$. In our setting we can always afford the time required to answer each query explicitly.

*Let $\mathcal{Q}$ be a set of counting queries, $k \in \mathbb{N}$ and $\alpha, \beta \in [0,1]$ be parameters. A generic sanitizer $\mathcal{M}$ is $(\alpha, \beta, \mathcal{Q}, k)$-*accurate *if for every database $D \in (\{0,1\}^d)^n$ and every sequence of queries $q_1, \ldots, q_k \in \mathcal{Q}^{(d)}$, $\mathcal{M}(D, \mathcal{Q})$ is α-accurate for D and $q_1, \ldots, q_k$ with probability at least $1 - \beta$ (over $\mathcal{M}$'s coins).*

If $\mathcal{M}$ is $(\alpha, \beta, \mathcal{Q}, k)$-accurate for any (constant) $\alpha < 1/2$ and $\beta = \beta(n) = o(1/n^2)$, we will drop α and β and say that $\mathcal{M}$ is $(\mathcal{Q}, k)$-accurate.

The parameters $\alpha < 1/2$, $\beta = o(1/n^2)$ are close to the weakest parameters possible, as a mechanism that answers 1/2 to every query achieves $\alpha = 1/2$, $\beta = 0$ for any number of arbitrary queries. That our lower bound applies to the parameters specified in Definition 2.2 makes our results stronger. Again, most positive results achieve (much) stronger accuracy parameters.

Efficiency of Sanitizers.

Simply, a sanitizer is efficient if it runs in time polynomial in the length of its input. To make the statement more precise, we need to specify how the queries are given to the sanitizer as input.

Notice that in order to specify an arbitrary counting query $q\colon \{0,1\}^d \to \{0,1\}$ requires 2^d bits. In this case, a sanitizer whose running time is polynomial in the time required to specify the query is not especially efficient. Thus, we restrict attention to queries that are efficiently computable, and have a succinct representation. In this work, we will fix the representation to be Boolean circuits over the basis $\{\wedge, \vee, \neg\}$ with possibly unbounded-fan-in. In this representation, any query can be evaluated in time $|q|$, where $|\cdot|$ denotes the size of the circuit computing q. We also want to consider the case where the queries are computable by circuits of low depth. For a constant $h \in \mathbb{N}$, we use $\mathcal{Q}^{(d)}_{\mathsf{depth}-h}$ to denote the set of all counting queries on $\{0,1\}^d$ specified by circuits of depth h. Finally, we use $\mathcal{Q}^{(d)}_{\mathsf{all}}$ to denote the set of all counting queries on $\{0,1\}^d$.

Definition 2.3. *A sanitizer $\mathcal{M}$ is* efficient *if, on input a database $D \in (\{0,1\}^d)^n$ and k queries $q_1, \ldots, q_k \in \mathcal{Q}^{(d)}_{\mathsf{all}}$, $\mathcal{M}$ runs in time* $\mathrm{poly}(d, n, k, |q_1| + \cdots + |q_k|)$. *For every $h \in \mathbb{N}$, a sanitizer $\mathcal{M}$ is* efficient for depth-h *queries if, on input a database $D \in (\{0,1\}^d)^n$ and k queries $q_1, \ldots, q_k \in \mathcal{Q}^{(d)}_{\mathsf{depth}-h}$, $\mathcal{M}$ runs in time* $\mathrm{poly}(d, n, k, |q_1| + \cdots + |q_k|)$.

For comparison with our results, we will recall the properties of some known mechanisms, stated in our terminology and for our choice of parameters:

Theorem 2.4 (Laplace Mechanism [6, 7]). *There exists a sanitizer $\mathcal{M}_{\mathsf{Lap}}$ that is 1) differentially private, 2) efficient, and 3) $(\mathcal{Q}^{(d)}_{\mathsf{all}}, \widetilde{\Omega}(n^2))$-accurate.*

Theorem 2.5 ([2, 8, 10, 13]). *There exists a sanitizer $\mathcal{M}_{\mathsf{Adv}}$ that is 1) differentially private and 2) $(\mathcal{Q}^{(d)}_{\mathsf{all}}, 2^{\widetilde{\Omega}(n/\sqrt{d})})$-accurate. For queries $q_1, \ldots, q_k \in \mathcal{Q}^{(d)}_{\mathsf{all}}$, $\mathcal{M}_{\mathsf{Adv}}$ runs in time* $\mathrm{poly}(2^d, n, k, |q_1| + \cdots + |q_k|)$.

These mechanisms can achieve stronger quantitative privacy and accuracy guarantees (in terms of ε, δ for privacy and α, β for accuracy) with only a small degradation in the number of queries.

3. TRAITOR-TRACING SCHEMES

In this section we give a definition of a traitor-tracing scheme, heavily tailored to the task of proving hardness results for generic sanitizers. We will sacrifice some consistency with the standard definitions. See below for further discussion of the ways in which our definition departs from the standard definition of traitor-tracing. In some cases, the non-standard aspects of the definition will be necessary to establish our results, and in others it will be for convenience. Despite these differences, we will henceforth refer to schemes satisfying our definition simply as *traitor-tracing schemes.*

3.1 Traitor-Tracing Schemes

First we describe the syntax of a traitor-tracing scheme more formally. For functions $n, k_{\mathsf{TT}} \colon \mathbb{N} \to \mathbb{N}$, an (n, k_{TT})-traitor-tracing scheme is a tuple of algorithms $(\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}}, \mathsf{Trace}_{\mathsf{TT}})$. We allow all the algorithms to be randomized except for $\mathsf{Dec}_{\mathsf{TT}}$.

- The algorithm $\mathsf{Gen}_{\mathsf{TT}}$ takes a security parameter, κ, and returns a sequence of $n = n(\kappa)$ user keys $\vec{sk} = (sk^{(1)}, \dots, sk^{(n)}) \in \{0,1\}^{\kappa}$. Formally, $\vec{sk} = (sk^{(1)}, \dots, sk^{(n)}) \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{TT}}(1^{\kappa})$.
- The algorithm $\mathsf{Enc}_{\mathsf{TT}}$ takes a sequence of n user keys $\vec{sk}$ and a message bit $b \in \{0,1\}$, and generates a ciphertext $c \in \mathcal{C} = \mathcal{C}^{(\kappa)}$. Formally, $c \leftarrow_{\mathrm{R}} \mathsf{Enc}_{\mathsf{TT}}(\vec{sk}, b)$.
- The algorithm $\mathsf{Dec}_{\mathsf{TT}}$ takes any single user key sk and a ciphertext $c \in \mathcal{C}$, runs in time $\mathrm{poly}(\kappa, n(\kappa))$ and deterministically returns a message bit $\widehat{b} \in \{0,1\}$. Formally $\widehat{b} = \mathsf{Dec}_{\mathsf{TT}}(sk, c)$.
- The algorithm $\mathsf{Trace}_{\mathsf{TT}}$ takes as input a set of user keys $\vec{sk} \in (\{0,1\}^{\kappa})^{n(\kappa)}$ and an oracle $\mathcal{P} \colon (\mathcal{C}^{(\kappa)})^{k_{\mathsf{TT}}(\kappa)} \to \{0,1\}^{k_{\mathsf{TT}}(\kappa)}$, makes only a single k_{TT}-tuple of queries, $(c_1, \dots, c_{k_{\mathsf{TT}}}) \in \mathcal{C}^{(\kappa)}$ to its oracle ($k_{\mathsf{TT}} = k_{\mathsf{TT}}(\kappa)$), and returns the name of a user $i \in [n(\kappa)]$. Formally, $i \leftarrow_{\mathrm{R}} \mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}}(\vec{sk})$.

Intuitively, think of the oracle $\mathcal{P}$ as being given some subset of keys $\vec{sk}_S = (sk^{(i)})_{i \in S}$ for a non-empty set $S \subseteq [n]$, and $\mathsf{Trace}_{\mathsf{TT}}$ is attempting to identify a user $i \in S$. Clearly, if $\mathcal{P}$ ignores its input and always returns 0, $\mathsf{Trace}_{\mathsf{TT}}$ cannot have any hope of success, so we must assume that $\mathcal{P}$ is capable of decrypting ciphertexts.

Definition 3.1. *Let Π_{TT} be an (n, k_{TT})-traitor-tracing scheme. Let $\mathcal{P}$ be a (possibly randomized) algorithm. We say that $\mathcal{P}$ is a* k_{TT}-available pirate decoder *if for every $\kappa \in \mathbb{N}$, every set of user keys $\vec{sk} = (sk^{(1)}, \dots, sk^{(n)}) \in \{0,1\}^{\kappa}$, every $S \subseteq [n]$ such that $|S| \geq n - 1$, and every $c_1, \dots, c_{k_{\mathsf{TT}}} \in \mathcal{C}^{(\kappa)}$, when $(\widehat{b}_1, \dots, \widehat{b}_{k_{\mathsf{TT}}}) \leftarrow_R \mathcal{P}(\vec{sk}_S, c_1, \dots, c_{k_{\mathsf{TT}}})$*

$$\Pr\left[\begin{array}{c}\exists j \in [k_{\mathsf{TT}}], b \in \{0,1\} \\ (\forall i \in S, \mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c_j) = b) \wedge \left(\widehat{b}_j \neq b\right)\end{array}\right] \leq o\left(\frac{1}{n(\kappa)^2}\right)$$

In other words, if every user key $sk^{(i)}$ (for $i \in S$) decrypts c to 1 (resp. 0), then $\mathcal{P}(\vec{sk}_S, \cdot)$ decrypts c to 1 (resp. 0), with high probability.

Definition 3.2. *Let Π_{TT} be an (n, k_{TT})-traitor-tracing scheme. Let $k_{\mathsf{TT}} \colon \mathbb{N} \to \mathbb{N}$ be a function. We say that Π_{TT} is a* secure (n, k_{TT})-traitor-tracing scheme *if for every $S \subseteq [n(\kappa)]$ such that $|S| \geq n(\kappa) - 1$, for every (possibly randomized) algorithm $\mathcal{P}$ that 1) runs in time* $\mathrm{poly}(\kappa, n(\kappa), k_{\mathsf{TT}}(\kappa))$ *and 2) is a k_{TT}-available pirate decoder, we have*

$$\Pr_{\substack{\vec{sk} \leftarrow_R \mathsf{Gen}_{\mathsf{TT}}(1^{\kappa}) \\ \mathcal{P}\text{'s, } \mathsf{Trace}_{\mathsf{TT}}\text{'s coins}}}\left[\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}(\vec{sk}_S, \cdot)}(\vec{sk}) \notin S\right] = o\left(\frac{1}{n(\kappa)}\right)$$

Remarks About Our Definition of Traitor-Tracing.

The traitor-tracing schemes we consider are somewhat different than those previously studied in the literature.

- We do not require the encryption or tracing algorithms to use public keys. In the typical application of traitor-tracing schemes to content distribution, these would be desirable features, however they are not necessary for proving hardness of sanitization.
- We only require that the tracing algorithm succeeds with probability $1 - o(1/n)$. Typically one would require that the tracing algorithm succeeds with probability $1 - n^{-\omega(1)}$.
- We do not give the pirate decoder access to an encryption oracle. In other words, we do not require CPA security. Most traitor-tracing schemes in the literature are public-key, making this distinction irrelevant. Here, we only need an encryption scheme that is secure for an *a priori* bounded number of messages.
- We allow the pirate decoder to be *stateful*, but in an unusual way. We require (roughly) that if any of the queries are ciphertexts generated by $\mathsf{Enc}(\vec{sk}, b)$, then the pirate decoder answers b to those queries, regardless of the other queries issued. In many models, the pirate is allowed to abort, and answer $\perp$ if it detects that it is being traced. However, we do allow our pirate to correlate its answers to different queries, subject to this accuracy constraint. We also allow the pirate to see all the queries made by the tracer at once, which is more power than is typically given to the pirate.

Roughly, the first three modifications will allow us to find a candidate scheme with very simple decryption and the fourth modification will allow us to trace stateful pirates even in the setting of bit-encryption.

3.2 Decryption Function Families

For Theorem 1.2, we are interested in traitor-tracing schemes where $\mathsf{Dec}_{\mathsf{TT}}$ is a "simple" function of the user key (for every ciphertext $c \in \mathcal{C}$).

Definition 3.3 (Decryption Function Family). *Let $(\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}})$ be a traitor-tracing scheme where $\mathsf{Gen}_{\mathsf{TT}}$ produces keys in $\{0,1\}^{\kappa}$ and $\mathsf{Enc}_{\mathsf{TT}}$ produce ciphertexts in $\mathcal{C} = \mathcal{C}^{(\kappa)}$. For every $c \in \mathcal{C}$, we define the* c-decryption function *$q_c \colon \{0,1\}^{\kappa} \to \{0,1\}$ to be $q_c(sk) = \mathsf{Dec}_{\mathsf{TT}}(sk, c)$. We define the* decryption function family $\mathcal{Q}^{(\kappa)}_{\mathsf{Dec}_{\mathsf{TT}}} = \{q_c\}_{c \in \mathcal{C}^{(\kappa)}}$.

In what follows, we will say that Π_{TT} is an traitor-tracing scheme with decryption function family $\mathcal{Q}^{(\kappa)}_{\mathsf{Dec}_{\mathsf{TT}}}$.

4. ATTACKING EFFICIENT SANITIZERS

In this section we will prove our main result, showing that the existence of traitor-tracing schemes (as in Definition 3.2) implies that efficient sanitizers cannot answer too many counting queries while satisfying differential privacy.

THEOREM 4.1. *Assume a traitor-tracing scheme,* $\Pi_{\mathsf{TT}} = (\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}}, \mathsf{Trace}_{\mathsf{TT}})$*, that is* $(n(\kappa), k_{\mathsf{TT}}(\kappa))$*-secure, with decryption function family* $\mathcal{Q}^{(\kappa)}_{\mathsf{Dec}_{\mathsf{TT}}}$*. Then there does not exist any sanitizer* $\mathcal{M}\colon (\{0,1\}^d)^n \times (\mathcal{Q}^{(d)}_{\mathsf{Dec}_{\mathsf{TT}}})^{k_{\mathsf{TT}}(d)} \to \mathbb{R}^{k_{\mathsf{TT}}(d)}$ *that is simultaneously 1) differentially private, 2) efficient, and 3)* $(\mathcal{Q}_{\mathsf{Dec}_{\mathsf{TT}}}, k_{\mathsf{TT}}(d))$*-accurate for* $\mathcal{Q}_{\mathsf{Dec}_{\mathsf{TT}}} = \cup_{d\in\mathbb{N}} \mathcal{Q}^{(d)}_{\mathsf{Dec}_{\mathsf{TT}}}$.

In the typical setting of parameters, we would have $n(\kappa) = \mathrm{poly}(\kappa)$, $k_{\mathsf{TT}}(\kappa) = \widetilde{\Theta}(n^2)$, and decryption can be implemented by circuits of size $\mathrm{poly}(n) = \mathrm{poly}(\kappa)$. In this setting, Theroem 4.1 says that there is no DP sanitizer $\mathcal{M}$ that takes a database $D \in (\{0,1\}^d)^{\mathrm{poly}(d)}$, runs in $\mathrm{poly}(d)$ time, and accurately answers $\widetilde{\Theta}(n^2)$ queries implemented by circuits of size $\mathrm{poly}(d)$.

We now sketch the proof: Every function $q_c \in \mathcal{Q}^{(d)}$ is viewed as a query $q_c(x)$ on a database row $x \in \{0,1\}^d$. Assume there is an efficient sanitizer is that is $(\mathcal{Q}^{(d)}_{\mathsf{Dec}_{\mathsf{TT}}}, k_{\mathsf{TT}}(d))$-accurate. The fact that $\mathcal{M}$ is accurate for these queries will imply that (after small modifications) $\mathcal{M}$ is a k_{TT}-available pirate decoder. Here is where we differ from Dwork et al., who assume that $\mathcal{M}$ accurately answers *all* queries in $\mathcal{Q}^{(d)}$, in which case $\mathcal{M}$ can be viewed as a stateless pirate decoder (but must solve a harder sanitization problem). We also differ in that we allow the traitor-tracing scheme to have the relaxed functionality and security discussed at the end of Section 3.

We complete the proof as in Dwork et al. Consider two experiments: In the first, we construct an n-row database D by running $\mathsf{Gen}_{\mathsf{TT}}(1^d)$ to obtain n user keys, and putting one in each row of D. Then we run $\mathsf{Trace}_{\mathsf{TT}}$ on $\mathcal{M}(D,\cdot)$ and obtain a user i. Since $\mathcal{M}$ is useful, and Π_{TT} is secure, we will have that $i \in [n]$ with probability close to 1, and thus there is an $i^* \in [n]$ such that $i = i^*$ with probability $\gtrsim 1/n$.

In the second experiment, we construct a database D' exactly as in the first, however we exclude the key $sk^{(i^*)}$. Since D and D' differ in only one row, differential privacy requires that $\mathsf{Trace}_{\mathsf{TT}}$, run with oracle $\mathcal{M}(D',\cdot)$, still outputs i^* with probability $\Omega(1/n)$. However, in this experiment, i^*, $sk^{(i^*)}$ is no longer given to the pirate decoder, and thus security of Π_{TT} says that $\mathsf{Trace}_{\mathsf{TT}}$, run with this oracle, must output i^* with probability $o(1/n)$, a contradiction.

PROOF OF THM 4.1. Assume we have a traitor-tracing scheme $\Pi_{\mathsf{TT}} = (\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}}, \mathsf{Trace}_{\mathsf{TT}})$ and assume there exists an efficient, differentially private, $(\mathcal{Q}^{(d)}, k_{\mathsf{TT}}(d))$-accurate sanitizer $\mathcal{M}$. We define the pirate decoder $\mathcal{P}_{\mathcal{M}}$ as follows: Since $\mathcal{M}$ is efficient, its running time is at most

Algorithm 1 The pirate decoder $\mathcal{P}_{\mathcal{M}}$

```
Input: A set of user keys $(\vec{sk}_S) \in \{0,1\}^d$ and a set of ciphertexts $c_1,\dots,c_{k_{\mathsf{TT}}}$ $(k_{\mathsf{TT}} = k_{\mathsf{TT}}(d))$.
Construct circuits for the queries $q_{c_1},\dots,q_{c_{k_{\mathsf{TT}}}} \in \mathcal{Q}^{(d)}_{\mathsf{Dec}_{\mathsf{TT}}}$.
Construct a database $D = (sk^{(i)})_{i\in S} \in (\{0,1\}^d)^{|S|}$.
Let $a_1,\dots,a_{k_{\mathsf{TT}}} \leftarrow_{\mathrm{R}} \mathcal{M}(D, q_{c_1},\dots,q_{c_{k_{\mathsf{TT}}}})$.
Round the answers $a_1,\dots,a_{k_{\mathsf{TT}}} \in [0,1]$ to obtain $\widehat{b}_1,\dots,\widehat{b}_{k_{\mathsf{TT}}} \in \{0,1\}$ (i.e. $\widehat{b}_j = \lceil a_j \rfloor$)
Output: $\widehat{b}_1,\dots,\widehat{b}_{k_{\mathsf{TT}}}$.
```

$\mathrm{poly}(d, n(d), k_{\mathsf{TT}}(d), |q_{c_1}| + \ldots + |q_{c_{k_{\mathsf{TT}}}}|)$, which is at most $\mathrm{poly}(d, n(d), k_{\mathsf{TT}}(d))$. Recall that the size of the circuits (queries) $q_c \in \mathcal{Q}^{(d)}_{\mathsf{Dec}_{\mathsf{TT}}}$ is $\mathrm{poly}(d,n)$. In this case $\mathcal{P}_{\mathcal{M}}$ runs in time $\mathrm{poly}(d, n(d), k_{\mathsf{TT}}(d))$ as well, since constructing the queries can be done in time polynomial in their size, and the final rounding step can be done in time $\mathrm{poly}(k_{\mathsf{TT}}(d))$.

Next, we claim that if $\mathcal{M}$ is accurate for $\mathcal{Q}^{(d)}$, then $\mathcal{P}_{\mathcal{M}}$ is a useful pirate decoder.

CLAIM 4.2. *If* $\mathcal{M}$ *is* $(\mathcal{Q}_{\mathsf{Dec}_{\mathsf{TT}}}, k_{\mathsf{TT}})$*-accurate, then* $\mathcal{P}_{\mathcal{M}}$ *is a* k_{TT}*-useful pirate decoder.*

PROOF OF CLAIM 4.2. Let $\vec{sk} \in \{0,1\}^d$ be a set of user keys for Π_{TT} and let $S \subseteq [n]$ be a subset of the users such that $|S| \geq n-1$. Suppose $c \in \mathcal{C}^{(d)}$ and $b \in \{0,1\}$ are such that for every $i \in S$, $\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c) = b$. Then we have that, for D as in $\mathcal{P}_{\mathcal{M}}$,

$$q_c(D) = \frac{1}{|S|}\sum_{i\in S} q_c(sk^{(i)}) = \frac{1}{|S|}\sum_{i\in S} \mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c) = b$$

Let $c_1,\dots,c_{k_{\mathsf{TT}}}$ be a set of ciphertexts, $q_{c_1},\dots,q_{c_{k_{\mathsf{TT}}}}$ and $a_1,\dots,a_{k_{\mathsf{TT}}}$ be as in $\mathcal{P}_{\mathcal{M}}$. The accuracy of $\mathcal{M}$ (with constant error $\alpha < 1/2$) guarantees that

$$\Pr\left[\exists j \in [k_{\mathsf{TT}}], \left|a_j - f_{c_j}(D)\right| \geq 1/2\right] = o(1/|S|^2)$$

Since $|S| \geq n-1$, $o(1/|S|^2) = o(1/n^2)$. Assuming $a_1,\dots,a_{k_{\mathsf{TT}}}$ is accurate up to error $\alpha < 1/2$ for $q_{c_1},\dots,q_{c_{k_{\mathsf{TT}}}}$, a_j will be rounded to exactly q_{c_j} whenever $q_{c_j}(D) \in \{0,1\}$. That is,

$$\Pr\left[\begin{array}{c}\exists j \in [k_{\mathsf{TT}}], b \in \{0,1\} \\ (\forall i \in S, \mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c_j) = b) \wedge \left(\widehat{b}_j \neq b\right)\end{array}\right] = o\left(\frac{1}{n(\kappa)^2}\right)$$

Thus, $\mathcal{P}_{\mathcal{M}}$ is k_{TT}-useful. This proves the claim. □

Since $\mathcal{P}_{\mathcal{M}}$ is a k_{TT}-useful pirate decoder, and Π_{TT} is a (n, k_{TT})-secure traitor-tracing scheme, running $\mathsf{Trace}_{\mathsf{TT}}$ on $\mathcal{P}_{\mathcal{M}}$ will always return some user $i \in [n]$. Thus there must be some user i^* that $\mathsf{Trace}_{\mathsf{TT}}$ returns with probability $\gtrsim 1/n$. Specifically, for every $\kappa \in \mathbb{N}$, there exists $i^*(\kappa) \in [n(\kappa)]$ such that,

$$\Pr_{\substack{\vec{sk} \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{TT}}(1^\kappa) \\ \mathcal{P}_{\mathcal{M}},\, \mathsf{Trace}_{\mathsf{TT}}}}\left[\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}_{\mathcal{M}}(\vec{sk},\cdot)}(\vec{sk}) = i^*\right] \geq \frac{1}{n(\kappa)} - o\left(\frac{1}{n(\kappa)}\right) \tag{1}$$

Let $S(\kappa) = [n(\kappa)] \setminus \{i^*(\kappa)\}$ Now we claim that if $\mathcal{M}$ is differentially private, then $\mathsf{Trace}_{\mathsf{TT}}$ will output $i^*(\kappa)$ with significant probability, even $\mathcal{P}_{\mathcal{M}}$ is not given the key of user $i^*(\kappa)$.

CLAIM 4.3. *If* $\mathcal{M}$ *is differentially private (for* $\varepsilon = O(1)$*,* $\delta = o(1/n)$*), then*

$$\Pr_{\substack{\vec{sk} \leftarrow_R \mathsf{Gen}_{\mathsf{TT}}(1^\kappa) \\ \mathcal{P}_{\mathcal{M}}\text{'s, } \mathsf{Trace}_{\mathsf{TT}}\text{'s coins}}}\left[\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}_{\mathcal{M}}(\vec{sk},\cdot)}(\vec{sk}) = i^*(\kappa)\right] \geq \Omega\left(\frac{1}{n(\kappa)}\right).$$

PROOF OF CLAIM 4.3. Fix any κ and let $k_{\mathsf{TT}} = k_{\mathsf{TT}}(\kappa)$ and $i^* = i^*(\kappa)$, $S = S(\kappa)$ as above. Let $D = \vec{sk}$ and $D_{-i^*} = \vec{sk}_S$. Take T to be the set of responses $\widehat{b}_1,\dots,\widehat{b}_{k_{\mathsf{TT}}}$ such that $\mathsf{Trace}_{\mathsf{TT}}(\vec{sk})$, after querying its oracle on ciphertexts $c_1,\dots,c_{k_{\mathsf{TT}}}$ and receiving responses $\widehat{b}_1,\dots,\widehat{b}_{k_{\mathsf{TT}}}$, outputs i^* (T depends on the coins of $\mathsf{Gen}_{\mathsf{TT}}$ and $\mathsf{Trace}_{\mathsf{TT}}$). By differential privacy, we have that

$$\Pr\left[\mathcal{M}(D, q_{c_1},\dots,q_{c_{k_{\mathsf{TT}}}}) \in T\right]$$
$$\leq e^{O(1)} \cdot \Pr\left[\mathcal{M}(D_{-i^*}, q_{c_1},\dots,q_{c_{k_{\mathsf{TT}}}}) \in T\right] + o\left(\frac{1}{n}\right).$$

Note that the queries constructed by $\mathcal{P}_{\mathcal{M}}$ depends only on $c_1, \ldots, c_{k_{\mathsf{TT}}}$, not on $\vec{sk}_S$. Also note that the final rounding step does not depend on the input at all. Thus, for every $T \subseteq \{0,1\}^{k_{\mathsf{TT}}}$

$$\Pr\left[\mathcal{P}_{\mathcal{M}}(\vec{sk}, c_1, \ldots, c_{k_{\mathsf{TT}}}) \in T\right]$$
$$\le e^{O(1)} \cdot \Pr\left[\mathcal{P}_{\mathcal{M}}(\vec{sk}_S, c_1, \ldots, c_{k_{\mathsf{TT}}}) \in T\right] + o\left(\frac{1}{n}\right).$$

The claim follows by combining with (1). □

To complete the proof, notice that the probability specified in Claim 4.3 is exactly the probability that $\mathsf{Trace}_{\mathsf{TT}}$ outputs the user i^*, when given the oracle $\mathcal{P}_{\mathcal{M}}(\vec{sk}_S)$, for $S = [n] \setminus \{i^*\}$. However, the fact that $\mathcal{P}_{\mathcal{M}}$ is efficient, and Π_{TT} is a secure traitor-tracing scheme implies that this probability is $o(1/n)$. Thus we have obtained a contradiction. This completes the proof of the Theorem.

5. TRAITOR-TRACING SCHEMES

In this section we show how to construct traitor-tracing schemes that satisfy Definition 3.2, and thus can be used to instantiate Theorem 4.1. First we will informally describe a simple construction that requires the tracing algorithm to make a sub-optimal number of queries, but will hopefully give the reader more intuition about the construction and how it differs from previous constructions of traitor-tracing schemes. Then we will give precise definitions of the encryption schemes (Section 5.2) and fingerprinting codes (Section 5.3) required for our construction. Then we will present the final construction more formally (Section 5.4). Due to space requirements, we omit the full security proof. Finally, we will use the weakened security requirements of the encryption scheme to show that our traitor-tracing scheme can be instantiated so that decryption is computable by constant-depth circuits (Section 5.6).

5.1 Outlining the Construction

Our construction is a variant of the most basic tracing traitor-tracing scheme [5]. Start with an encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Generate an independent key $sk^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Gen}$ for each user (we will ignore the security parameter in the informal description). To encrypt $b \in \{0,1\}$, encrypt it under each user's key independently and concatenate the ciphertexts. That is,

$$\mathsf{Enc}_{\mathsf{TT}}(\vec{sk}, b) = (\mathsf{Enc}(sk^{(1)}, b), \ldots, \mathsf{Enc}(sk^{(n)}, b)).$$

Each user can decrypt the ciphertext by applying Dec, as long as she knows which part of the ciphertext to decrypt.

Now we describe how an available pirate decoder for this scheme can be traced. As with all traitor-tracing schemes, we will form ciphertexts that different users would decrypt differently, assuming they decrypt as intended using the algorithm $\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, \cdot)$. We can do so with the following algorithm:

$$\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i) = (\mathsf{Enc}(sk^{(1)}, 1), \ldots,$$
$$\mathsf{Enc}(sk^{(i)}, 1), \mathsf{Enc}(sk^{(i+1)}, 0), \ldots, \mathsf{Enc}(sk^{(n)}, 0)$$

for $i = 0, 1, \ldots, n$. The algorithm forms a ciphertext that users $1, \ldots, i$ will decrypt to 1 and users $i+1, \ldots, n$ will decrypt to 0.

The tracing algorithm generates a random sequence of indices $i_1, \ldots, i_{k_{\mathsf{TT}}} \in \{0, 1, \ldots, n\}$, for $k_{\mathsf{TT}} = (n+1)s$, such that each element of $\{0, 1, \ldots, n\}$ appears exactly s times, where s is a parameter to be chosen later. Then, for every j it generates a ciphertext $c_j \leftarrow_{\mathrm{R}} \mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i_j)$. Next, it queries $\mathcal{P}_{\vec{sk}_S}(c_1, \ldots, c_{k_{\mathsf{TT}}})$. Given the output of the pirate, the tracing algorithm computes $P_i = \frac{1}{s}\sum_{j: i_j = i} \mathcal{P}(\vec{sk}, c_1, \ldots, c_{k_{\mathsf{TT}}})_j$ for $i = 0, 1, \ldots, n$. Finally, the tracing algorithm outputs any i^* such that $P_{i^*} - P_{i^*-1} \ge 1/n$.

Now we explain why this algorithm successfully traces efficient available pirate decoders. Notice that if we choose c according to $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, 0)$, then every user decrypts c to 0, so $P_0 = 0$. Similarly, $P_n = 1$. Thus, there exists i^* such that $P_{i^*} - P_{i^*-1} \ge 1/n$. Next, we argue that i^* is in S except with small probability. Notice that $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i^*)$ and $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i^*-1)$ differ only in the message encrypted under key $sk^{(i^*)}$, so if $i^* \notin S$, this key is unknown to the pirate decoder. The security of the encryption scheme (made precise in Definition 5.2) guarantees that if $sk^{(i^*)}$ is unknown to an efficient pirate, then we can replace k_{TT} uses of $\mathsf{Enc}(sk^{(i^*)}, 1)$ with $\mathsf{Enc}(sk^{(i^*)}, 0)$, and this change will only affect the success probability of the pirate by $o(1/n)$. But after we make this replacement, $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i^*)$ and $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i^*-1)$ are (perfectly, information-theoretically) indistinguishable to the pirate. Since the sequence of indices $i_1, \ldots, i_{k_{\mathsf{TT}}}$ is random, the pirate has no information about which elements i_j are i^* and which are i^*-1. Thus, if the pirate wants to make P_{i^*} larger than P_{i^*-1}, for some $i^* \notin S$, she can do no better than to "guess". If we take $s = \widetilde{O}(n^2)$, and apply a Chernoff bound, it turns out that for every $i \notin S$, $P_i - P_{i-1} = o(1/n)$. This conclusion also holds after we take into account the security loss of the encryption scheme, which is $o(1/n)$. Thus, the scheme we described is a secure traitor-tracing scheme in the sense of Definition 3.2.

In arguing that the scheme is secure, we used the fact that $P_0 = 0$ and $P_n = 1$ *no matter what other queries are made to the pirate*. In many applications, this assumption would not be reasonable. However, when the pirate is derived from an accurate sanitizer, this condition will be satisfied.

For this scheme, the tracer makes $(n+1)s = \widetilde{O}(n^3)$ queries. Before proceeding, we will explain how to reduce the number of queries from $\widetilde{O}(n^3)$ to $\widetilde{O}(n^2)$. The high-level argument that the scheme is secure used two facts:

1. By the availability of the pirate decoder, if every user in S would decrypt a ciphertext c to b, then the pirate decrypts c to b (in the above, $P_0 = 0, P_n = 1$).

2. Because of the encryption, a pirate decoder without user i's key "doesn't know" how user i would decrypt each ciphertext.

Systems leveraging these two properties to identify a colluding user are called *fingerprinting codes* [4], and have been studied extensively. In fact, the tracing algorithm we described is identical to the tracing algorithm we define in Section 5.4, but instantiated with the fingerprinting code of Boneh and Shaw [4], which has length $\widetilde{O}(n^3)$. Tardos [18] constructed shorter fingerprinting codes, with length $\widetilde{O}(n^2)$, which we use to reduce the number of queries to trace.

Before giving the final construction, we a precise definition of the security we need from an encryption scheme, and then we will give a formal definition of fingerprinting codes.

5.2 Encryption Schemes

An encryption scheme is tuple of three efficient algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. All the algorithms may be randomized except for Dec. The scheme has the following syntax:

- The algorithm Gen takes a security parameter κ, runs in time $\mathrm{poly}(\kappa)$, and returns a private key $sk \in \{0,1\}^{\kappa}$. Formally $sk \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^{\kappa})$.
- The algorithm Enc takes a private key and a message bit $b \in \{0,1\}$ and generates a ciphertext $c \in \mathcal{C} = \mathcal{C}^{(\kappa)}$. Formally, $c \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk, b)$.
- The algorithm Dec takes a private key $sk \in \{0,1\}^{\kappa}$ and a ciphertext $c \in \mathcal{C}^{(\kappa)}$, runs in time $\mathrm{poly}(\kappa)$, and returns a message bit $\widehat{b}$.

Definition 5.1. *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* (perfectly) correct *if for every* $b \in \{0,1\}$*, and every* $\kappa \in \mathbb{N}$, $\Pr_{sk \leftarrow_R \mathsf{Gen}(1^{\kappa})}[\mathsf{Dec}(sk, \mathsf{Enc}(sk, b)) = b] = 1$.

We require that our schemes have the following k_{Enc}-message security property.

Definition 5.2. *Let* $\varepsilon_{\mathsf{Enc}} \colon \mathbb{N} \to [0,1]$ *and* $k_{\mathsf{Enc}} \colon \mathbb{N} \to \mathbb{N}$, $\mathrm{T}_{\mathsf{Enc}} \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *be functions. An encryption scheme* Π_{Enc} *is* $(\varepsilon_{\mathsf{Enc}}, k_{\mathsf{Enc}}, \mathrm{T}_{\mathsf{Enc}})$-secure *if for every* $\mathrm{T}_{\mathsf{Enc}}(\kappa, k_{\mathsf{Enc}}(\kappa))$*-time algorithm* $\mathcal{A}_{\mathsf{Enc}}$ *and every two sequences* $b = (b_1, \ldots, b_{k_{\mathsf{Enc}}}), b' = (b'_1, \ldots, b'_{k_{\mathsf{Enc}}}) \in \{0,1\}$ *(for* $k_{\mathsf{Enc}} = k_{\mathsf{Enc}}(\kappa)$*),*

$$\left| \Pr_{sk \leftarrow_R \mathsf{Gen}(1^{\kappa})}[\mathcal{A}_{\mathsf{Enc}}(c_1, \ldots, c_{k_{\mathsf{Enc}}}) = 1] - \Pr_{sk \leftarrow_R \mathsf{Gen}(1^{\kappa})}\left[\mathcal{A}_{\mathsf{Enc}}(c'_1, \ldots, c'_{k_{\mathsf{Enc}}}) = 1\right] \right| \leq \varepsilon_{\mathsf{Enc}}(\kappa).$$

where $c_j \leftarrow_R \mathsf{Enc}(sk, b_j)$ *and* $c'_j \leftarrow_R \mathsf{Enc}(sk, b'_j)$.

Notice that we do not require Π_{Enc} to be secure against adversaries that are given $\mathsf{Enc}(sk, \cdot)$ as an oracle. That is, we do not require CPA security.

Definition 5.3 (Encryption Scheme). *We say that a tuple of algorithms* Π_{Enc} *is an* $(\varepsilon_{\mathsf{Enc}}, k_{\mathsf{Enc}}, \mathrm{T}_{\mathsf{Enc}})$*-encryption scheme if is correct and* $(\varepsilon_{\mathsf{Enc}}, k_{\mathsf{Enc}}, \mathrm{T}_{\mathsf{Enc}})$*-secure.*

5.3 Fingerprinting Codes

As we alluded to above, our tracing algorithm will use a *fingerprinting code*, introduced by Boneh and Shaw [4]. A fingerprinting code is a pair of efficient (possibly randomized) algorithms $(\mathsf{Gen}_{\mathsf{FP}}, \mathsf{Trace}_{\mathsf{FP}})$ with the following syntax.

- The algorithm $\mathsf{Gen}_{\mathsf{FP}}$ takes a number of users n as input and outputs a codebook of n codewords of length $\ell_{\mathsf{FP}} = \ell_{\mathsf{FP}}(n)$, $W = (w^{(1)}, \ldots, w^{(n)}) \in \{0,1\}^{\ell_{\mathsf{FP}}}$. Formally $W \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{FP}}(1^n)$. We will think of $W \in \{0,1\}^{n \times \ell_{\mathsf{FP}}}$ as a matrix with each row containing a codeword.
- The algorithm $\mathsf{Trace}_{\mathsf{FP}}$ takes an n-user codebook W and a word $w' \in \{0,1\}^{\ell_{\mathsf{FP}}}$ and returns an index $i \in [n]$. Formally, $i = \mathsf{Trace}_{\mathsf{FP}}(W, w')$.

Given a non-empty subset $S \subseteq [n]$ and a set of codewords $W_S = (w^{(i)})_{i \in S} \in \{0,1\}^{\ell_{\mathsf{FP}}}$, we define the set of *feasible codewords* to be $F(W_S) = \left\{ w' \mid \forall j \in [\ell_{\mathsf{FP}}]\, \exists i \in S\; w'_j = w_j^{(i)} \right\}$. Informally, if all users in S have a 0 (resp. 1) in the j-th symbol of their codeword, then they must produce a word with 0 (resp. 1) as the j-th symbol. We also define the *critical positions* to be the set of indices for which this constraint is binding. That is,

$$\mathrm{Crit}(W_S) = \left\{ j \in [\ell_{\mathsf{FP}}] \mid \exists b_j \in \{0,1\} \forall i \in S\; w_j^{(i)} = b_j \right\}.$$

The security of a fingerprinting code asserts that an adversary who is given a subset W_S of the codewords should not be able to produce an element of $F(W_S)$ that does not trace to a user $i \in S$. More formally,

Definition 5.4 (Secure Fingerprinting Code). *Let* $\varepsilon_{\mathsf{FP}} \colon \mathbb{N} \to [0,1]$ *and* $\ell_{\mathsf{FP}} \colon \mathbb{N} \to \mathbb{N}$ *be functions. A pair of algorithms* $(\mathsf{Gen}_{\mathsf{FP}}, \mathsf{Trace}_{\mathsf{FP}})$ *is an* $(\varepsilon_{\mathsf{FP}}, \ell_{\mathsf{FP}})$-fingerprinting code *if* $\mathsf{Gen}_{\mathsf{FP}}(1^n)$ *outputs a codebook* $W \in \{0,1\}^{n \times \ell_{\mathsf{FP}}(n)}$*, and furthermore, for every (possibly inefficient) algorithm* $\mathcal{A}_{\mathsf{FP}}$*, and every non-empty* $S \subseteq [n]$,

$$\Pr_{W \leftarrow_R \mathsf{Gen}_{\mathsf{FP}}(1^n)}\left[\begin{array}{c} w' \leftarrow_R \mathcal{A}_{\mathsf{FP}}(W_S) \\ w' \in F(W_S) \wedge \mathsf{Trace}_{\mathsf{FP}}(W, w') \notin S \end{array} \right] \leq \varepsilon_{\mathsf{FP}}(n)$$

where the two executions of $\mathcal{A}_{\mathsf{FP}}$ *are understood to be the same.*

The following theorem is due to Tardos [18].

Theorem 5.5 ([18]). *For every function* $\varepsilon_{\mathsf{FP}} \colon \mathbb{N} \to [0,1]$*, there exists a* $\left(o(1/n^2), O(n^2 \log n)\right)$*-finger-printing code.*

5.4 The Traitor-Tracing Scheme

We are now ready to state the construction more formally (Algorithms 2 & 3)

Algorithm 2 The traitor-tracing scheme Π_{TT} (without its tracing algorithm).

```
Let an encryption $\Pi_{\mathsf{Enc}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a function
$n \colon \mathbb{N} \to \mathbb{N}$ be parameters of the scheme. Assume that
$n(\kappa) \leq 2^{\kappa/2}$ for every $\kappa \in \mathbb{N}$

$\mathsf{Gen}_{\mathsf{TT}}(1^{\kappa})$:
For every user $i = 1, \ldots, n(\kappa)$, let $\overline{sk}^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^{\kappa/2})$
Let $sk^{(i)} = (\overline{sk}^{(i)}, i)$ (padded with zeros to have length
exactly $\kappa$).
Output $\vec{sk} = (sk^{(1)}, \ldots, sk^{(n)})$
(We will sometimes use $sk^{(i)}$ and $\overline{sk}^{(i)}$ interchangeably)

$\mathsf{Enc}_{\mathsf{TT}}(sk^{(1)}, \ldots, sk^{(n)}, b)$:
For every user $i$, let $c^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk^{(i)}, b)$
Output $c = (c^{(1)}, \ldots, c^{(n)})$

$\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c)$:
Output $\widehat{b} = \mathsf{Dec}(sk^{(i)}, c^{(i)})$
```

5.5 Security of Π_{TT}

In this section we sketch why our construction is secure. It can be verified from the specification of the scheme that it has the desired syntactic properties, that it generates $n(\kappa)$ user keys, and that the tracing algorithm makes $\ell_{\mathsf{FP}}(n(\kappa))$ non-adaptive queries to its oracle.

To argue traceability, as in the sketch (Section 5.1), we want to generate a set of ciphertexts that different users decrypt in different ways. Specifically, given a fingerprinting code $W \in \{0,1\}^{n \times \ell_{\mathsf{FP}}}$ (represented as a matrix with $w^{(i)}$ in the i-th row), we want to generate a set of ciphertexts $c_1, \dots, c_{\ell_{\mathsf{FP}}}$, such that user i, if she decrypts as intended using $\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, \cdot)$, will decrypt c_j to $w_j^{(i)}$. That is, $\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c_j) = w_j^{(i)}$. $\mathsf{Trace}_{\mathsf{TT}}$ will query the pirate decoder on these ciphertexts, treat these responses as a word w', run the tracing algorithm for the fingerprinting code on w', and use the output of $\mathsf{Trace}_{\mathsf{FP}}$ as its own output.

If $\mathcal{P}$ is available, its output will be a feasible codeword for W_S. To see this, recall that if every user $i \in S$ decrypts c_j to the same bit, then an available pirate decoder $\mathcal{P}(\vec{sk}_S, \cdot)$, decrypts c_j to that bit. However, the critical positions of W_S are exactly those for which every user $i \in S$ has the same symbol in position j. Thus, the codeword returned by the pirate is feasible, and the fingerprinting code's tracing algorithm can identify a user in S.

Algorithm 3 The tracing algorithm for Π_{TT} and a subroutine $\mathsf{TrEnc}_{\mathsf{TT}}$.

```
Let a length $\ell_{\mathsf{FP}} = \ell_{\mathsf{FP}}(n)$ fingerprinting code $\Pi_{\mathsf{FP}} = (\mathsf{Gen}_{\mathsf{FP}}, \mathsf{Trace}_{\mathsf{FP}})$ be a parameter of the scheme and let $\Pi_{\mathsf{Enc}}$ be as in Algorithm 2.

$\mathsf{TrEnc}_{\mathsf{TT}}(sk^{(1)}, \dots, sk^{(n)}, W)$:
Let $n \times k$ be the dimensions of $W$
For every $i \in [n], j \in [k]$, let $c_j^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk^{(i)}, W_{i,j})$
For every $j \in [k]$, let $c_j = (c_j^{(1)}, \dots, c_j^{(n)})$
Output $c_1, \dots, c_k$
(Notice that $\mathsf{Dec}(sk^{(i)}, c_j^{(i)}) = W_{i,j}$)

$\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}}(\vec{sk})$:
Let $n$ be the number of user keys and $\ell_{\mathsf{FP}} = \ell_{\mathsf{FP}}(n)$
Let $W \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{FP}}(1^n)$
Let $\widehat{b}_1, \dots, \widehat{b}_{\ell_{\mathsf{FP}}} \leftarrow_{\mathrm{R}} \mathcal{P}(\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, W))$ and let $w' = \widehat{b}_1 \| \dots \| \widehat{b}_{\ell_{\mathsf{FP}}}$
Output $i \leftarrow_{\mathrm{R}} \mathsf{Trace}_{\mathsf{FP}}(W, w')$
```

The catch in this argument is that $\mathsf{TrEnc}_{\mathsf{TT}}$ takes all of W as input, however an attacker for the fingerprinting code is only allowed to see W_S, and thus cannot simulate $\mathsf{TrEnc}_{\mathsf{TT}}$ in a security reduction. However, if $\mathcal{P}$ only has keys $\vec{sk}_S$, and $i \notin S$, then an efficient $\mathcal{P}$ cannot decrypt the i-th component of a ciphertext $c = (c^{(1)}, \dots, c^{(n)})$. But these are the only components that depend on $w^{(i)}$. So $w^{(i)}$ is computationally hidden from $\mathcal{P}$ anyway, and we could replace that codeword with a string of zeros without significantly affecting the success probability of $\mathcal{P}$. Formalizing this intuition will yield a valid attacker for the fingerprinting code, and obtain a contradiction.

Theorem 5.6. *Let Π_{Enc} be an $(\varepsilon_{\mathsf{Enc}}, k_{\mathsf{Enc}}, \mathrm{T}_{\mathsf{Enc}})$-secure encryption scheme, and Π_{FP} be a $(\varepsilon_{\mathsf{FP}}, \ell_{\mathsf{FP}})$-fingerprinting code, Π_{FP}. Let $n, k_{\mathsf{TT}} \colon \mathbb{N} \to \mathbb{N}$ be any functions such that for every $\kappa \in \mathbb{N}$, $n(\kappa) \le 2^{\kappa/2}$ and*

1. *the encryption scheme and fingerprinting code have sufficiently strong security, $n(\kappa) \cdot \varepsilon_{\mathsf{Enc}}(\kappa) + \varepsilon_{\mathsf{FP}}(n(\kappa)) = o\left(\frac{1}{n(\kappa)^2}\right)$,*
2. *the encryption scheme is secure for sufficiently many queries, $k_{\mathsf{Enc}}(\kappa) \ge k_{\mathsf{TT}}(\kappa) = \ell_{\mathsf{FP}}(n(\kappa))$,*
3. *the encryption scheme is secure against adversaries whose running time is as long as the pirate decoder's, for every $a > 0$, $\mathrm{T}_{\mathsf{Enc}}(\kappa/2, k_{\mathsf{TT}}(\kappa)) \ge (\kappa + n(\kappa) + k_{\mathsf{TT}}(\kappa))^a$.*

Then Π_{TT} instantiated with Π_{Enc} and Π_{FP} is an (n, k_{TT})-traitor-tracing scheme.

Due to space limitations, we defer the full proof—which follows this intuition very closely—to the full version.

5.6 Decryption Function Family of Π_{TT}

Recall that the two goals of constructing a new traitor-tracing scheme were to trace stateful pirates and to reduce the complexity of decryption. We addressed tracing of stateful pirates in the previous section, and now we turn to the complexity of decryption. We do so by instantiating the traitor-tracing scheme with various encryption schemes and making two observations: 1) The type of encryption schemes we require are sufficiently weak that there already exist plausible candidates with a very simple decryption operation, and 2) Decryption for the traitor-tracing scheme is not much more complex than decryption for the underlying encryption scheme. We summarize the second point with the following simple lemma.

Lemma 5.7 (Decryption Func. Family for Π_{TT}). *Let Π_{TT} be as defined, with Π_{Enc} as its underlying encryption scheme. Let $(\overline{sk}, i) = sk \in \{0,1\}^\kappa$ and $c = (c^{(1)}, \dots, c^{(n)}) \in \mathcal{C}^{(\kappa)}$ be any user key and ciphertext for Π_{TT}. Then*

$$\mathsf{Dec}_{\mathsf{TT},c}(sk) = \mathsf{Dec}_{\mathsf{TT},c}(\overline{sk}, i) = \bigvee_{i' \in [n]} \left(\mathbf{1}_{i'}(i) \wedge \mathsf{Dec}_{c^{(i')}}(\overline{sk})\right)$$

Here, the function $\mathbf{1}_x(y)$ takes the value 1 if $y = x$ and 0 otherwise. The lemma follows directly from the construction of $\mathsf{Dec}_{\mathsf{TT}}$. Also note that the function $\mathbf{1}_{i'} \colon \{0,1\}^{\lceil \log n \rceil} \to \{0,1\}$ is just a conjunction of $\lceil \log n \rceil$ bits (a single gate of fan-in $O(\log n)$), and we need to compute n of these functions. In addition to computing $\mathbf{1}_{i'}$ and $\mathsf{Dec}_{c^{(i')}}$, there are n conjunctions and a single outer disjunction. Thus we add an additional $n+1$ gates, compute decryption n times, and increase the depth by 2. Hence, an intuitive summary of the lemma is that if Dec can be implemented by circuits of size s and depth h, $\mathsf{Dec}_{\mathsf{TT}}$ can be implemented by circuits of size $n \cdot (s + O(\log n)) = \widetilde{O}(ns)$ and depth $h+2$. This summary will be precise enough to state our main results.

By combining Lemma 5.7 with Theorem 5.6, we easily obtain the following corollary.

Corollary 5.8. *Let $n = n(\kappa)$ be any polynomial in κ. Assuming the existence of (non-uniformly secure) one-way functions, there exists an $(n, \widetilde{O}(n^2))$-secure traitor-tracing scheme with decryption function family $\mathcal{Q}_{\mathsf{Dec}_{\mathsf{TT}}}^{(\kappa)}$ consisting only of circuits of size* $\mathrm{poly}(\kappa)$

Theorem 1.1 in the introduction follows by combining Theorem 4.1 with Corollary 5.8.

By Lemma 5.7, in order to construct a traitor-tracing scheme whose decryption can be computed by constant-depth circuits, it is sufficient to find an encryption scheme where decryption can be implemented by constant-depth circuits. A scheme meeting the relaxed security criteria of Definition 5.2 can be constructed from a sufficiently good *local pseudorandom generator (PRG)*. A recent result of Applebaum [1] gave the first plausible candidate construction of a local PRG for the range of parameters we need, giving plausibility to the assumption that such PRGs (and, as we show, traitor-tracing schemes with constant-depth decryption) exist. We note that local PRGs actually imply encryption schemes with local decryption, which is stronger than just constant-depth decryption.

Definition 5.9 (Local PRG). *An efficient algorithm* $\mathsf{G}\colon \{0,1\}^{\kappa} \to \{0,1\}^{s_{\mathsf{PRG}}(\kappa)}$ *is a* $\varepsilon_{\mathsf{PRG}}$*-pseudorandom generator if for every* $\mathrm{poly}(s_{\mathsf{PRG}}(\kappa))$*-time adversary* $\mathcal{A}_{\mathsf{PRG}}$

$$\left|\Pr\left[\mathcal{A}_{\mathsf{PRG}}(\mathsf{G}(U_\kappa)) = 1\right] - \Pr\left[\mathcal{A}_{\mathsf{PRG}}(U_{s_{\mathsf{PRG}}(\kappa)}) = 1\right]\right| \le \varepsilon_{\mathsf{PRG}}(\kappa)$$

If, in addition, if each bit of the output depends only on some set of L *bits of the input, then* G *is a* $(\varepsilon_{\mathsf{PRG}}, L)$*-local pseudorandom generator.*

It is a well known result in Cryptography that pseudorandom generators imply encryption schemes satisfying Definition 5.2 (for certain ranges of parameters). We will use a particular construction whose decryption can be computed in constant-depth whenever the underlying PRG is locally-computable (or, more generally, computable by constant-depth circuits). The construction is the standard "computational one-time pad", which we state below for concreteness.

Algorithm 4 An encryption scheme Π_{LocalEnc} that can be decrypted in constant depth.

```
$\mathsf{Gen}(1^\kappa)$:
    Let $s \leftarrow_{\mathrm{R}} \{0,1\}^\kappa$ and output $sk = s$
$\mathsf{Enc}(sk, b)$:
    Let $r \leftarrow_{\mathrm{R}} \{1, 2, \ldots, s_{\mathsf{PRG}}(\kappa)\}$, output $c = (r, \mathsf{G}(sk)_r \oplus b)$
$\mathsf{Dec}(sk, c)$:
    Let $(r', b') = c$ and output: $b = \mathsf{G}(sk)_r \oplus b'$
```

Lemma 5.10 (Local PRGs → Encryption). *If there exists a* $(\varepsilon_{\mathsf{PRG}}(\kappa), L)$*-local PRG,* $\mathsf{G}\colon \{0,1\}^\kappa \to \{0,1\}^{s_{\mathsf{PRG}}(\kappa)}$*, then there exists an* $(\varepsilon_{\mathsf{Enc}} = \varepsilon_{\mathsf{PRG}} + k_{\mathsf{Enc}}^2/s_{\mathsf{PRG}}, k_{\mathsf{Enc}})$*-Secure Encryption Scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with decryption func. family* $\mathcal{Q}_{\mathsf{Dec},\kappa}$ *consisting of circuits of size* $\mathrm{poly}(\kappa)$ *and depth* 4.

We defer the proof of Lemma 5.10 to the full version of this work. Proving security is straightforward using standard arguments in Cryptography. The proof that decryption can be computed in constant depth follows easily from expressing decryption of a ciphertext $c = (r, b)$ in the form

$$\mathsf{Dec}_{(r,b)}(s) = \bigvee_{i \in [s_{\mathsf{PRG}}(\kappa)]} \left(\mathbf{1}_i(r) \wedge (\mathsf{G}_i(s) \oplus b)\right).$$

This scheme yields the following corollary.

Corollary 5.11. *Let* $n = n(\kappa)$ *be any polynomial in* κ*. Assuming the existence of a* $(o(1/n^2), n^7, L)$*-local pseudorandom generator for some constant* $L \in \mathbb{N}$*, there exists an* $(n, \widetilde{O}(n^2))$*-secure traitor-tracing scheme with decryption function family* $\mathcal{Q}^{(\kappa)}_{\mathsf{Dec}_{\mathsf{TT}}}$ *consisting of circuits of size* $\widetilde{O}(n) \cdot \mathrm{poly}(\kappa)$ *and depth* 6.

Theorem 1.2 in the introduction follows by combining Theorem 4.1 with Corollary 5.11.

6. REFERENCES

[1] Applebaum, B. Pseudorandom generators with long stretch and low locality from random local one-way functions. In *STOC* (2012), pp. 805–816.

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

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

[4] Boneh, D., and Shaw, J. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory 44*, 5 (1998), 1897–1905.

[5] Chor, B., Fiat, A., and Naor, M. Tracing traitors. In *CRYPTO* (1994), pp. 257–270.

[6] Dinur, I., and Nissim, K. Revealing information while preserving privacy. In *PODS* (2003).

[7] Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC '06* (2006), pp. 265–284.

[8] Dwork, C., Naor, M., Reingold, O., Rothblum, G. N., and Vadhan, S. P. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC '09* (2009).

[9] Dwork, C., Naor, M., and Vadhan, S. The privacy of the analyst and the power of the state. *Manuscript* (2012).

[10] Dwork, C., Rothblum, G. N., and Vadhan, S. P. Boosting and differential privacy. In *FOCS* (2010), IEEE Computer Society, pp. 51–60.

[11] Gupta, A., Hardt, M., Roth, A., and Ullman, J. Privately releasing conjunctions and the statistical query barrier. In *STOC '11* (2011), pp. 803–812.

[12] Gupta, A., Roth, A., and Ullman, J. Iterative constructions and private data release. In *TCC* (2012), pp. 339–356.

[13] Hardt, M., Ligett, K., and McSherry, F. A simple and practical algorithm for differentially private data release. *NIPS '12* (2012).

[14] Hardt, M., and Rothblum, G. N. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS* (2010).

[15] Hardt, M., Rothblum, G. N., and Servedio, R. A. Private data release via learning thresholds. In *SODA* (2012), pp. 168–187.

[16] Kiayias, A., and Yung, M. On crafty pirates and foxy tracers. In *Digital Rights Management Workshop* (2001), pp. 22–39.

[17] Roth, A., and Roughgarden, T. Interactive privacy via the median mechanism. In *STOC '10* (2010), pp. 765–774.

[18] Tardos, G. Optimal probabilistic fingerprint codes. *J. ACM 55*, 2 (2008).

[19] Thaler, J., Ullman, J., and Vadhan, S. P. Faster algorithms for privately releasing marginals. In *ICALP (1)* (2012), pp. 810–821.

[20] Ullman, J. Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. *CoRR abs/1207.6945* (2012).

[21] Ullman, J., and Vadhan, S. P. PCPs and the hardness of generating private synthetic data. In *TCC '11* (2011), pp. 400–416.