Hardness of Non-Interactive Differential Privacy from One-Way Functions

Lucas Kowalczyk[*] Tal Malkin[†] Jonathan Ullman[‡] Daniel Wichs[§]

December 22, 2017

Abstract

A central challenge in differential privacy is to design computationally efficient non-interactive algorithms that can answer large numbers of *statistical queries* on a sensitive dataset. That is, we would like to design a differentially private algorithm that takes a dataset $D \in X^n$ consisting of some small number of elements n from some large data universe X, and efficiently outputs a summary that allows a user to efficiently obtain an answer to any query in some large family Q.

Ignoring computational constraints, this problem can be solved even when X and Q are exponentially large and n is just a small polynomial; however, all algorithms with remotely similar guarantees run in exponential time. There have been several results showing that, under the strong assumption of indistinguishability obfuscation (iO), no efficient differentially private algorithm exists when X and Q can be exponentially large. However, there are no strong separations between information-theoretic and computationally efficient differentially private algorithms under any standard complexity assumption.

In this work we show that, if one-way functions exist, there is no general purpose differentially private algorithm that works when X and Q are exponentially large, and n is an arbitrary polynomial. In fact, we show that this result holds even if X is just subexponentially large (assuming only polynomially-hard one-way functions). This result solves an open problem posed by Vadhan in his recent survey [Vad16].

[*]Columbia University Department of Computer Science. `luke@cs.columbia.edu`.

[†]Columbia University Department of Computer Science. `tal@cs.columbia.edu`.

[‡]Northeastern University College of Computer and Information Science. `jullman@ccs.neu.edu`.

[§]Northeastern University College of Computer and Information Science. `wichs@ccs.neu.edu`

1 Introduction

A central challenge in privacy research is to generate rich private *summaries* of a sensitive dataset. Doing so creates a tension between two competing goals. On one hand we would like to ensure *differential privacy* [DMNS06]—a strong notion of individual privacy that guarantees no individual's data has a significant influence on the summary. On the other hand, the summary should enable a user to obtain approximate answers to some large set of queries. Since the summary must be generated without knowing which queries the user will need to answer, we would like Q to be very large. This problem is sometimes called *non-interactive query release*, in contrast with *interactive query release* where the user is required specify the (much smaller) set of queries that he needs to answer in advance, and the private answers may be tailored to just these queries.

More specifically, there is a sensitive dataset $D = (D_1, \ldots, D_n) \in X^n$ where each element of D is the data of some individual, and comes from some *data universe* X. We are interested in generating a summary that allows the user to answer *statistical queries* on D, which are queries of the form "What fraction of the individuals in the dataset satisfy some property q?" [Kea93]. Given a set of statistical queries Q and a data universe X, we would like to design a differentially private algorithm M that takes a dataset $D \in X^n$ and outputs a summary that can be used to obtain an approximate answer to every query in Q. Since differential privacy requires hiding the information single individual, for a fixed (X, Q) generating a private summary becomes easier as n becomes larger. The overarching goal is to find algorithms that are both private and accurate for X and Q as large as possible and n as small as possible.

Since differential privacy is a strong guarantee, *a priori* we might expect differentially private algorithms to be very limited. However, a seminal result of Blum, Ligett, and Roth [BLR13] showed how to generate a differentially private summary encoding answers to *exponentially many* queries. After a series of improvements and extensions [DNR+09, DRV10, RR10, HR14, GRU12, HLM12, NTZ16, Ull15], we know that any set of queries Q over any universe X can be answered given a dataset of size $n \gtrsim \sqrt{\log|X|} \cdot \log|Q|$ [HR14]. Thus, it is information-theoretically possible to answer huge sets of queries using a small dataset.

Unfortunately, all of these algorithms have running time $\mathrm{poly}(n, |X|, |Q|)$, which can be exponential in the *dimension* of the dataset, and in the *description* of a query. For example if $X = \{0,1\}^d$, so each individual's data consists of d binary attributes, then the dataset has size nd but the running time will be at least 2^d. Thus, these algorithms are only efficient when both $|X|$ and $|Q|$ have polynomial size. There are computationally efficient algorithms when one of $|Q|$ and $|X|$ is very large, provided that the other is extremely small—at most $n^{2-\Omega(1)}$. Specifically, (1) the classical technique of perturbing the answer to each query with independent noise requires a dataset of size $n \gtrsim \sqrt{|Q|}$ [DN03, DN04, BDMN05, DMNS06], and (2) the folklore *noisy histogram algorithm* (see e.g. [Vad16]) requires a dataset of size $n \gtrsim \sqrt{|X| \cdot \log|Q|}$. Thus there are huge gaps between the power of information-theoretic and computationally efficient differentially private algorithm.

Beginning with the work of Dwork et al. [DNR+09], there have been a series of results giving evidence that this gap is inherent using a connection to *traitor-tracing schemes* [CFN94]. The first such result by Dwork et al. [DNR+09] showed the first separation between efficient and inefficient differentially private algorithms, proving a polynomial-factor separation in sample complexity between the two cases assuming bilinear cryptography. Subsequently, Boneh and Zhandry [BZ14] proved that, under the much stronger assumption that indistinguishability obfuscation (iO) exists, then a worst-case family of statistical queries, there is no computationally efficient algorithm with $\mathrm{poly}(\log|Q| + \log|X|)$ sample complexity. Even more recently, Kowalczyk et al. [KMUZ16] strengthened these results to show that the two efficient algorithms mentioned above—independent perturbation and the noisy histogram—are optimal up to polynomial factors, also assuming iO.

Reference	Data Universe $\lvert X_\kappa \rvert$	# of Queries $\lvert Q_\kappa \rvert$	Dataset Size $n(\kappa)$	Assumption
[DNR+09, BSW06]	$\geq \exp(\kappa)$	$\geq \exp(\kappa)$	$\leq \kappa^{2-\Omega(1)}$	Bilinear Maps
[DNR+09, BZ14]	$\geq \exp(\kappa)$	$\geq \exp(\kappa)$	$\leq \mathrm{poly}(\kappa)$	iO + OWF
[KMUZ16]	$\geq \exp(\kappa)$	$\geq \tilde{O}(n^7)$	$\leq \mathrm{poly}(\kappa)$	iO + OWF
[KMUZ16]	$\geq \tilde{O}(n^7)$	$\geq \exp(\kappa)$	$\leq \mathrm{poly}(\kappa)$	iO + OWF
This work	$\geq \exp(\kappa^{o(1)})$	$\geq \exp(\kappa)$	$\leq \mathrm{poly}(\kappa)$	OWF

Table 1: Comparison of Hardness Results for Offline Differentially Private Query Release. Each row corresponds to an informal statement of the form "If the assumption holds, then there is no general purpose differentially private algorithm that works when the data universe has size at least $|X|$, the number of queries is at least $|Q|$, and the size of the dataset is at most n." All assumptions are polynomial-time hardness.

These results give a relatively clean picture of the complexity of non-interactive differential privacy, but only if we assume the existence of iO. Recently, in his survey on the foundations of differential privacy [Vad16], Vadhan posed it as an open question to prove hardness of non-interactive differential privacy using standard cryptographic assumptions. In this work, we resolve this open question by proving a strong hardness result for non-interactive differential privacy making only the standard assumption that one-way functions (OWF) exist.

Theorem 1.1. *There is a sequence of pairs $\{(X_\kappa, Q_\kappa)\}_{\kappa\in\mathbb{N}}$ where $|X_\kappa| = 2^{2^{\mathrm{poly}(\log\log\kappa)}} = 2^{\kappa^{o(1)}}$ and $|Q_\kappa| = 2^\kappa$ such that, assuming the existence of one-way functions, for every polynomial $n = n(\kappa)$, there is no polynomial time differentially private algorithm that takes a dataset $D \in X_\kappa^n$ and outputs an accurate answer to every query in Q_κ up to an additive error of $\pm 1/3$.*

We remark that, in addition to removing the assumption of iO, Theorem 1.1 is actually stronger than that of Boneh and Zhandry [BZ14], since the data universe size can be subexponential in κ, even if we only make standard polynomial-time hardness assumptions. We leave it as an interesting open question to obtain quantitatively optimal hardness results matching (or even improving) those of [KMUZ16] using standard assumptions.

Like all of the aforementioned hardness results, the queries constructed in Theorem 1.1 are somewhat complex, and involve computing some cryptographic functionality. A major research direction in differential privacy has been to construct efficient non-interactive algorithms for specific large families of *simple* queries, or prove that this problem is hard. The main technique for constructing such algorithms has been to leverage efficient *PAC learning* algorithms. Specifically, a series of works [BLR13, GHRU13, GRU12, HRS12] have shown that an efficient PAC learning algorithm for a class of concepts related to Q can be used to obtain efficient differentially private algorithms for answering the queries Q. Thus, hardness results for differential privacy imply hardness results for PAC learning. However, it is relatively easy to show the hardness of PAC learning using just OWFs [PV88], and one can even show the hardness of learning simple concept classes (e.g. DNF formulae [DLS14, DSS16]) by using more structured complexity assumptions. One roadblock to proving hardness results for privately answering simple families of queries is that, prior to our work, even proving hardness results for worst-case families of queries required using extremely powerful cryptographic primitives like iO, leaving little room to utilize more structured complexity assumptions to obtain hardness for simple queries. By proving hardness results for differential privacy using only the assumption of one-way functions, we believe our results are an important step towards proving hardness results for simpler families of queries.

Relationship to [GKW17]. A concurrent and independent work by Goyal, Koppula, and Waters also shows how to prove hardness results for non-interactive differential privacy from weaker assumptions than iO. Specifically, they propose a new primitive called *risky traitor tracing* that has weaker security than standard traitor tracing, but is still strong enough to rule out the existence of computationally efficient differentially private algorithms, and construct such schemes under certain assumptions on composite-order bilinear maps. Unlike our work, their new primitive has applications outside of differential privacy. However, within the context of differential privacy, Theorem 1.1 is stronger than what they prove in two respects: (1) their bilinear-map assumptions are significantly stronger than our assumption of one-way functions, and (2) their hardness result requires a data universe of size $|X_\kappa| = \exp(\kappa)$, rather than our result, which allows $|X_\kappa| = \exp(\kappa^{o(1)})$.

1.1 Techniques

Differential Privacy and Traitor-Tracing Schemes. Our results build on the connection between differentially private algorithms for answering statistical queries and *traitor-tracing schemes*, which was discovered by Dwork et al. [DNR+09]. Traitor-tracing schemes were introduced by Chor, Fiat, and Naor [CFN94] 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 differential privacy. Very informally, they argue as follows. Suppose a coalition of users takes their keys and builds a dataset $D \in X^n$ where each element of the dataset contains one of their user keys. The family Q will contain a query q_c for each possible ciphertext c. The query q_c asks "What fraction of the elements (user keys) in D would decrypt the ciphertext c to the message 1?" Every user can decrypt, so if the sender encrypts a message $b \in \{0,1\}$ as a ciphertext c, then every user will decrypt c to b. Thus, the answer to the statistical query q_c will be b. Now, suppose there were an efficient algorithm that outputs an accurate answer to each query q_c in Q. Then the coalition could use it to efficiently produce a summary of the dataset 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 dataset from the summary, then the summary is not private.

Hardness of Privacy from OWF. In order to instantiate this outline, we need a sufficiently good traitor-tracing scheme. Traitor-tracing schemes can be constructed from any *functional encryption scheme for comparison functions* [BSW06][1] This is a cryptographic scheme in which secret keys are associated with functions f and ciphertexts are associated with a message x, and decrypting the ciphertext with a secret key corresponding to f reveals $f(x)$ and "nothing else." In our application, the functions are of the form f_z where $f_z(x) = 1$ if and only if $x \geq z$ (as integers).

[1]These were called *private linear broadcast encryption schemes* by [BSW06], but we use the more modern terminology of functional encryption.

Using techniques from [KMUZ16], we show that, in order to prove hardness results for differentially private algorithms it suffices to have a functional encryption scheme for comparison functions that is non-adaptively secure for just two ciphertexts and n secret keys. That is, if an adversary chooses to receive keys for n functions $f_1, \dots, f_n$, and ciphertexts for two messages x_1, x_2, then he learns nothing more than $\{f_i(x_1), f_i(x_2)\}_{i \in [n]}$. Moreover, the comparison functions only need to support inputs in $\{0, 1, \dots, n\}$ (i.e. $\log n$ bits). Lastly, it suffices for us to have a symmetric-key functional encryption scheme where both the encryption and key generation can require a private master secret key.

We then construct this type of functional encryption (FE) using the techniques of Gorbunov, Vaikuntanathan and Wee [GVW12] who constructed bounded-collusion FE from any public-key encryption. There are two important differences between the type of FE that we need and bounded-collusion FE in [GVW12]: (1) we want a symmetric-key FE based on one-way functions (OWFs), whereas they constructed public-key FE using public-key encryption, (2) we want security for only 2 ciphertexts but many secret keys, whereas they achieved security for many ciphertexts but only a small number of secret keys. It turns out that their construction can be rather easily scaled down from the public-key to the symmetric-key setting by replacing public-key encryption with symmetric-key encryption. Going from many ciphertexts and few secret keys to many secret keys and few ciphertexts essentially boils down to exchanging the role of secret keys and ciphertexts in their scheme, but this requires care. We give the full description and analysis of this construction. Lastly, we rely on one additional property: for the simple functions we consider with logarithmic input length, we can get a scheme where the ciphertext size is extremely small $\kappa^{o(1)}$, where κ is the security parameter, while being able to rely on standard polynomial hardness of OWFs. To do so, we replace the garbled circuits used in the construction of [GVW12] with information-theoretic randomized encodings for simple functions and leverage the fact that we are in more restrictive nonadaptive secret-key setting. The resulting small ciphertext size allows us to get DP lower bounds even when the data universe is of size $|X| = \exp(\kappa^{o(1)})$.

Why Two-Ciphertext Security? In the hardness reduction sketched above, the adversary for the functional encryption scheme will use the efficient differentially private algorithm to output some stateless program (the summary) that correctly decrypts ciphertexts for the functional encryption scheme (by approximately answering statistical queries). The crux of the proof is to use differential privacy to argue that the scheme must violate security of the functional encryption scheme by distinguishing encryptions of the messages x and $x-1$ even if it does not possess a secret key for the function f_x, which is the only function in the family of comparison functions that would give different output on these two messages, and therefore an adversary without this key should not be able to distinguish between these two messages.

Thus, in order to obtain a hardness result for differential privacy we need a functional encryption scheme with the following non-standard security definition: for every polynomial time adversary that obtains a set of secret keys corresponding to functions other than f_x and outputs some stateless program, with high probability that program has small advantage in distinguishing encryptions of x from $x-1$. Implicit in the work of Kowalczyk et al. [KMUZ16] is a lemma that says that this property is satisfied by any functional encryption scheme that satisfies the standard notion of security for two messages. At a high level, security for one-message allow for the possibility that the adversary sometimes outputs a program with large positive advantage and sometimes outputs a program with large negative advantage, whereas two-message security bounds the average *squared advantage*, meaning that the advantage must be small with high probability. This argument is similar to one used by Dodis and Yu [DY13] in a completely different setting.

1.2 Additional Related Work

(Hardness of) Interactive Differential Privacy. Another area of focus is *interactive* differential privacy, where the mechanism gets the dataset D and a (relatively small) set of queries Q chosen by the analyst and must output answers to each query in Q. Most differentially private algorithms for answering a large number of arbitrary queries actually work in this setting [DNR+09, DRV10, HLM12], or even in a more challenging setting where the queries in Q arrive online and may be adaptively chosen. [RR10, RR10, GRU12, Ull15]. Ullman [Ull16] showed that, assuming one-way functions exist, there is no polynomial-time differentially private algorithm that takes a dataset $D \in X^n$ and a set of $\tilde{O}(n^2)$ arbitrary statistical queries and outputs an accurate answer to each of these queries. The hardness of interactive differential privacy has also been extended to a seemingly easier model of *interactive data analysis* [HU14, SU15], which is closely related to differential privacy [DFH+15, BNS+16], even though privacy is not an explicit requirement in that model. These results however do not give any specific set of queries Q that can be privately summarized information-theoretically but not by a computationally efficient algorithm, and thus do not solve the problem addressed in thus work.

The Complexity of Simple Statistical Queries. As mentioned above, a major open research direction is to design non-interactive differentially private algorithms for simple families of statistical queries. For example, there are polynomial time differentially private algorithms with polynomial sample complexity for summarizing *point queries* and *threshold queries* [BNS13, BNSV15], using an information-theoretically optimal number of samples. Another class of focus has been marginal queries [GHRU13, HRS12, TUV12, CTUW14, DNT14]. A marginal query is defined on the data universe $\{0,1\}^\kappa$. It is specified by a set of positions $S \subseteq \{1,\ldots,\kappa\}$, and a pattern $t \in \{0,1\}^{|S|}$ and asks "What fraction of elements of the dataset have each coordinate $j \in S$ set to t_j?" Specifically, Thaler et al. [TUV12], building on the work of Hardt et al. [HRS12] gave an efficient differentially private algorithm for answering all marginal queries up to an additive error of $\pm.01$ when the dataset is of size $n \gtrsim 2^{\sqrt{\kappa}}$. If we assume sufficiently hard one-way functions exist, then Theorem 1.1 would show that these parameters are not achievable for an arbitrary set of queries. It remains a central open problem in differential privacy to either design an optimal computationally efficient algorithm for marginal queries or to give evidence that this problem is hard.

Hardness of Synthetic Data. There have been several other attempts to explain the accuracy vs. computation tradeoff in differential privacy by considering restricted classes of algorithms. For example, Ullman and Vadhan [UV11] (building on Dwork et al. [DNR+09]) show that, assuming one-way functions, no differentially private and computationally efficient algorithm that outputs a *synthetic dataset* can accurately answer even the very simple family of 2-way marginals. A synthetic dataset is a specific type of summary that is interchangeable with the real dataset—it is a set $\hat{D} = (\hat{D}_1,\ldots,\hat{D}_n) \in X^n$ such that the answer to each query on $\hat{D}$ is approximately the same as the answer to the same query on D. 2-way marginals are just the subset of marginal queries above where we only allow $|S| \leq 2$, and these queries capture the mean covariances of the attributes. This result is incomparable to ours, since it applies to a very small and simple family of statistical queries, but only applies to algorithms that output synthetic data.

Information-Theoretic Lower Bounds. A related line of work [BUV14, DTTZ14, BST14, BU17, SU17] uses ideas from *fingerprinting codes* [BS98] to prove information-theoretic lower bounds on the number of queries that can be answered by differentially private algorithms, and also devise realistic attacks against the privacy of algorithms that attempt to answer too many queries [DSS+15, DSSU17]. Most relevant to this work is the result of [BUV14] which says that if the size of the data universe is 2^{n^2}, then there is a fixed set of n^2 queries that no differentially private algorithm, even a computationally unbounded one, can answer accurately. Although these results are orthogonal to ours, the techniques are quite related, as fingerprinting codes are essentially the information-theoretic analogue of traitor-tracing schemes.

2 Differential Privacy Preliminaries

2.1 Differentially Private Algorithms

A *dataset* $D \in X^n$ is an ordered set of n rows, where each row corresponds to an individual, and each row is an element of some the *data universe* X. We write $D = (D_1, \dots, D_n)$ where D_i is the i-th row of D. We will refer to n as the *size* of the dataset. We say that two datasets $D, D' \in X^*$ are *adjacent* if D' can be obtained from D by the addition, removal, or substitution of a single row, and we denote this relation by $D \sim D'$. In particular, if we remove the i-th row of D then we obtain a new dataset $D_{-i} \sim D$. Informally, an algorithm A is differentially private if it is randomized and for any two adjacent datasets $D \sim D'$, the distributions of $A(D)$ and $A(D')$ are similar.

Definition 2.1 (Differential Privacy [DMNS06])**.** Let $A : X^n \to S$ be a randomized algorithm. We say that A is (ε, δ)-*differentially private* if for every two adjacent datasets $D \sim D'$ and every $E \subseteq S$,

$$\mathbb{P}[A(D) \in E] \leq e^{\varepsilon} \cdot \mathbb{P}[A(D') \in E] + \delta.$$

In this definition, ε, δ may be functions n.

2.2 Algorithms for Answering Statistical Queries

In this work we study algorithms that answer *statistical queries* (which are also sometimes called *counting queries*, *predicate queries*, or *linear queries* in the literature). For a data universe X, a statistical query on X is defined by a predicate $q : X \to \{0,1\}$. Abusing notation, we define the evaluation of a query q on a dataset $D = (D_1, \dots, D_n) \in X^n$ to be

$$\frac{1}{n} \sum_{i=1}^{n} q(D_i).$$

A single statistical query does not provide much useful information about the dataset. However, a sufficiently large and rich set of statistical queries is sufficient to implement many natural machine learning and data mining algorithms [Kea93], thus we are interesting in differentially private algorithms to answer such sets. To this end, let $Q = \{q : X \to \{0,1\}\}$ be a set of statistical queries on a data universe X.

Informally, we say that a mechanism is accurate for a set Q of statistical queries if it answers every query in the family to within error $\pm\alpha$ for some suitable choice of $\alpha > 0$. Note that $0 \leq q(D) \leq 1$, so this definition of accuracy is meaningful when $\alpha < 1/2$.

Before we define accuracy, we note that the mechanism may represent its answer in any form. That is, the mechanism outputs may output a *summary* $S \in \mathcal{S}$ that somehow represents the answers to every query in Q. We then require that there is an *evaluator* $Eval : \mathcal{S} \times \mathcal{Q} \to [0,1]$ that takes the summary and a query and outputs an approximate answer to that query. That is, we think of $Eval(S, q)$ as the mechanism's answer to the query q. We will abuse notation and simply write $q(S)$ to mean $Eval(S, q)$.[2]

[2]If we do not restrict the running time of the algorithm, then it is without loss of generality for the algorithm to

Definition 2.2 (Accuracy)**.** For a family Q of statistical queries on X, a dataset $D \in X^n$ and a summary $S \in \mathcal{S}$, we say that S *is α-accurate for Q on D* if

$$\forall q \in Q \quad |q(D) - q(S)| \leq \alpha.$$

For a family of statistical queries Q on X, we say that an algorithm $A : X^n \to S$ is (α, β)*-accurate for Q given a dataset of size n* if for every $D \in X^n$,

$$\mathbb{P}[A(D) \text{ is } \alpha\text{-accurate for } Q \text{ on } X] \geq 1 - \beta.$$

In this work we are typically interested in mechanisms that satisfy the very weak notion of $(1/3, o(1/n))$-accuracy, where the constant $1/3$ could be replaced with any constant $< 1/2$. Most differentially private mechanisms satisfy quantitatively much stronger accuracy guarantees. Since we are proving hardness results, this choice of parameters makes our results stronger.

2.3 Computational Efficiency

Since we are interested in asymptotic efficiency, we introduce a computation parameter $\kappa \in \mathbb{N}$. We then consider a sequence of pairs $\{(X_\kappa, Q_\kappa)\}_{\kappa \in \mathbb{N}}$ where Q_κ is a set of statistical queries on X_κ. We consider databases of size n where $n = n(\kappa)$ is a polynomial. We then consider algorithms A that take as input a dataset X_κ^n and output a summary in S_κ where $\{S_\kappa\}_{\kappa \in \mathbb{N}}$ is a sequence of output ranges. There is an associated evaluator Eval that takes a query $q \in Q_\kappa$ and a summary $s \in S_\kappa$ and outputs a real-valued answer. The definitions of differential privacy and accuracy extend straightforwardly to such sequences.

We say that such an algorithm is *computationally efficient* if the running time of the algorithm and the associated evaluator run in time polynomial in the computation parameter κ. In principle, it could require at many as $|X|$ bits even to specify a statistical query, in which case we cannot hope to answer the query efficiently, even ignoring privacy constraints. Thus, we restrict attention to statistical queries that are specified by a circuit of size $\mathrm{polylog}|X|$, and thus can be evaluated in time $\mathrm{polylog}|X|$, and so are not the bottleneck in computation. To remind the reader of this fact, we will often say that Q is a family of *efficiently computable statistical queries*.

2.4 Notational Conventions

Given a boolean predicate P, we will write $\mathbb{I}\{P\}$ to denote the value 1 if P is true and 0 if P is false. We also say that a function $\varepsilon = \varepsilon(n)$ is *negligible* if $\varepsilon(n) = O(1/n^c)$ for every constant $c > 0$, and denote this by $\varepsilon(n) = \mathrm{negl}(n)$.

3 Weakly Secure Traitor-Tracing Schemes

In this section we describe a very relaxed notion of traitor-tracing schemes whose existence will imply the hardness of differentially private data release.

simply output a list of real-valued answers to each queries by computing $Eval(S, q)$ for every $q \in Q$. However, this transformation makes the running time of the algorithm at least $|Q|$. The additional generality of this framework allows the algorithm to run in time sublinear in $|Q|$. This generality is crucial for our results, which apply to settings where the family of queries is superpolynomially large in the size of the dataset.

3.1 Syntax and Correctness

For a function $n : \mathbb{N} \to \mathbb{N}$ and a sequence $\{K_\kappa, C_\kappa\}_{\kappa \in \mathbb{N}}$, an $(n, \{K_\kappa, C_\kappa\})$-*traitor-tracing scheme* is a tuple of efficient algorithms $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax.

- Setup takes as input a security parameter κ, runs in time $\mathrm{poly}(\kappa)$, and outputs $n = n(\kappa)$ secret *user keys* $\mathsf{sk}_1, \dots, \mathsf{sk}_n \in K_\kappa$ and a secret *master key* msk. We will write $\vec{\mathsf{sk}} = (\mathsf{sk}_1, \dots, \mathsf{sk}_n, \mathsf{msk})$ to denote the set of keys.

- Enc takes as input a master key msk and an *index* $i \in \{0, 1, \dots, n\}$, and outputs a ciphertext $c \in C_\kappa$. If $c \leftarrow_{\mathrm{R}} \mathsf{Enc}(j, \mathsf{msk})$ then we say that c is *encrypted to index* j.

- Dec takes as input a ciphertext c and a user key sk_i and outputs a single bit $b \in \{0, 1\}$. We assume for simplicity that Dec is deterministic.

Correctness of the scheme asserts that if $\vec{\mathsf{sk}}$ are generated by Setup, then for any pair i, j, $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{Enc}(\mathsf{msk}, j)) = \mathbb{I}\{i \le j\}$. For simplicity, we require that this property holds with probability 1 over the coins of Setup and Enc, although it would not affect our results substantively if we required only correctness with high probability.

Definition 3.1 (Perfect Correctness). An $(n, \{K_\kappa, C_\kappa\})$-traitor-tracing scheme is *perfectly correct* if for every $\kappa \in \mathbb{N}$, and every $i, j \in \{0, 1, \dots, n\}$

$$\mathop{\mathbb{P}}_{\vec{\mathsf{sk}} = \mathsf{Setup}(\kappa),\, c = \mathsf{Enc}(\mathsf{msk}, j)} [\mathsf{Dec}(\mathsf{sk}_i, c) = \mathbb{I}\{i \le j\}] = 1.$$

3.2 Index-Hiding Security

Intuitively, the security property we want is that any computationally efficient adversary who is missing one of the user keys sk_{i^*} cannot distinguish ciphertexts encrypted with index i^* from index $i^* - 1$, even if that adversary holds all $n - 1$ other keys sk_{-i^*}. In other words, an efficient adversary cannot infer anything about the encrypted index beyond what is implied by the correctness of decryption and the set of keys he holds.

More precisely, consider the following two-phase experiment. First the adversary is given every key except for sk_{i^*}, and outputs a decryption program S. Then, a challenge ciphertext is encrypted to either i^* or to $i^* - 1$. We say that the traitor-tracing scheme is secure if for every polynomial time adversary, with high probability over the setup and the decryption program chosen by the adversary, the decryption program has small advantage in distinguishing the two possible indices.

Definition 3.2 (Index Hiding). A traitor-tracing scheme Π satisfies *(weak) index-hiding security* if for every sufficiently large $\kappa \in \mathbb{N}$, every $i^* \in [n(\kappa)]$, and every $\mathrm{poly}(\kappa)$-time adversary A,

$$\mathop{\mathbb{P}}_{\vec{\mathsf{sk}} = \mathsf{Setup}(\kappa),\, S = A(\mathsf{sk}_{-i^*})} \left[\mathbb{P}[S(\mathsf{Enc}(\mathsf{msk}, i^*)) = 1] - \mathbb{P}[S(\mathsf{Enc}(\mathsf{msk}, i^* - 1)) = 1] > \frac{1}{4en} \right] \le \frac{1}{4en} \tag{1}$$

In the above, the inner probabilities are taken over the coins of Enc and S.

Note that in the above definition we have fixed the success probability of the adversary for simplicity. Moreover, we have fixed these probabilities to relatively large ones. Requiring only a polynomially small advantage is crucial to achieving the key and ciphertext lengths we need to obtain our results, while still being sufficient to establish the hardness of differential privacy.

3.3 Index-Hiding Security Implies Hardness for Differential Privacy

It was shown by Kowalczyk et al. [KMUZ16] (refining similar results from [DNR+09, Ull16]) that a traitor-tracing scheme satisfying index-hiding security implies a hardness result for

Theorem 3.3. *Suppose there is an $(n, \{K_\kappa, C_\kappa\})$-traitor-tracing scheme that satisfies perfect correctness (Definition 3.1) and index-hiding security (Definition 3.2). Then there is a sequence of of pairs $\{X_\kappa, Q_\kappa\}_{\kappa \in \mathbb{N}}$ where Q_κ is a set of statistical queries on X_κ, $|Q_\kappa| = |C_\kappa|$, and $|X_\kappa| = |K_\kappa|$ such that there is no algorithm A that is simultaneously*

1. *computationally efficient,*
2. *$(1, 1/4n)$-differentially private, and*
3. *$(1/3, 1/2n)$-accurate for Q_κ on datasets $D \in X_\kappa^{n(\kappa)}$.*

3.4 Two-Index-Hiding-Security

While Definition 3.2 is the most natural to prove hardness of privacy, it is not consistent with the usual security definition for functional encryption because of the nested "probability-of-probabilities." In order to apply more standard notions of functional encryption, we show that index-hiding security follows from a more natural form of security for two ciphertexts.

First, consider the following IndexHiding game.

The challenger generates keys $\vec{\mathsf{sk}} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{msk}) \leftarrow_{\mathrm{R}} \mathsf{Setup}(\kappa)$.
The adversary A is given keys sk_{-i^*} and outputs a decryption program S.
The challenger chooses a bit $b \leftarrow_{\mathrm{R}} \{0, 1\}$
The challenger generates an encryption to index $i^* - b$, $c \leftarrow_{\mathrm{R}} \mathsf{Enc}(\mathsf{msk}, i^* - b)$
The adversary makes a guess $b' = S(c)$

Figure 1: $\mathrm{IndexHiding}_{i^*}$

Let $\mathrm{IndexHiding}_{i^*, \vec{\mathsf{sk}}, S}$ be the game $\mathrm{IndexHiding}_{i^*}$ where we fix the choices of $\vec{\mathsf{sk}}$ and S. Also, define

$$\mathrm{Adv}_{i^*, \vec{\mathsf{sk}}, S} = \mathop{\mathbb{P}}_{\mathsf{IndexHiding}_{i^*, \vec{\mathsf{sk}}, S}} [b' = b] - \frac{1}{2}.$$

so that

$$\mathop{\mathbb{P}}_{\mathsf{IndexHiding}_{i^*}} [b' = b] - \frac{1}{2} = \mathop{\mathbb{E}}_{\substack{\vec{\mathsf{sk}} = \mathsf{Setup}(\kappa) \\ S = A(\mathsf{sk}_{-i^*})}} \left[\mathrm{Adv}_{i^*, \vec{\mathsf{sk}}, S} \right]$$

Then the following statement implies (1) in Definition 3.2:

$$\mathop{\mathbb{P}}_{\vec{\mathsf{sk}} = \mathsf{Setup}(\kappa),\, S = A(\mathsf{sk}_{-i^*})} \left[\mathrm{Adv}_{i^*, \vec{\mathsf{sk}}, S} > \frac{1}{4en} \right] \leq \frac{1}{2en} \tag{2}$$

We can define a related two-index-hiding game.

The challenger generates keys $\vec{\mathsf{sk}} = (\mathsf{sk}_1, \dots, \mathsf{sk}_n, \mathsf{msk}) \leftarrow_{\mathrm{R}} \mathsf{Setup}$.
The adversary A is given keys sk_{-i^*} and outputs a decryption program S.
Choose $b_0 \leftarrow_{\mathrm{R}} \{0,1\}$ and $b_1 \leftarrow_{\mathrm{R}} \{0,1\}$ independently.
Let $c_0 \leftarrow_{\mathrm{R}} \mathsf{Enc}(i^* - b_0; \mathsf{msk})$ and $c_1 \leftarrow_{\mathrm{R}} \mathsf{Enc}(i^* - b_1; \mathsf{msk})$.
Let $b' = S(c_0, c_1)$.

Figure 2: $\mathrm{TwoIndexHiding}_{i^*}$

Analogous to what we did with IndexHiding, we can define $\mathrm{TwoIndexHiding}_{i^*,\vec{\mathsf{sk}},S}$ to be the game $\mathrm{TwoIndexHiding}_{i^*}$ where we fix the choices of $\vec{\mathsf{sk}}$ and S, and define

$$\mathrm{TwoAdv}_{i^*} = \underset{\mathrm{TwoIndexHiding}_{i^*}}{\mathbb{P}}[b' = b_0 \oplus b_1] - \frac{1}{2}$$

Kowalczyk et al. [KMUZ16] proved the following lemma that will be useful to connect our new construction to the type of security definition that implies hardness of differential privacy.

Lemma 3.4. *Let Π be a traitor-tracing scheme such that for every efficient adversary A, every $\kappa \in \mathbb{N}$, and index $i^* \in [n(\kappa)]$,*

$$\mathrm{TwoAdv}_{i^*} \le \frac{1}{300n^3}$$

Then Π satisfies weak index-hiding security.

In the rest of the paper, we will construct a scheme satisfying the assumption of the above lemma with suitable key and ciphertexts lenghts, which we can immediately plug into Theorem 3.3 to obtain Theorem 1.1 in the introduction.

4 Cryptographic Tools

4.1 Decomposable Randomized Encodings

Let $\mathcal{F} = \left\{ f : \{0,1\}^\ell \to \{0,1\}^k \right\}$ be a family of Boolean functions. An *(information-theoretic) decomposable randomized encoding for $\mathcal{F}$* is a pair of efficient algorithms $(\mathsf{DRE.Encode}, \mathsf{DRE.Decode})$ such that the following hold:

- $\mathsf{DRE.Encode}$ takes as input a function $f \in \mathcal{F}$ and randomness R and outputs a randomized encoding consisting of a set of ℓ pairs of labels

$$\tilde{F}(f,R) = \left\{ \begin{array}{ccc} \tilde{F}_1(f,0,R) & \cdots & \tilde{F}_\ell(f,0,R) \\ \tilde{F}_1(f,1,R) & \cdots & \tilde{F}_\ell(f,1,R) \end{array} \right\}$$

  where the i-th pair of labels corresponds to the i-th bit of the input x.

- **(Correctness)** $\mathsf{DRE.Decode}$ takes as input a set of ℓ labels corresponding to some function f and input x and outputs $f(x)$. Specifically,

$$\forall\ f \in \mathcal{F},\ x \in \{0,1\}^\ell \quad \mathsf{DRE.Decode}\Big(\tilde{F}_1(f,x_1,R), \dots, \tilde{F}_\ell(f,x_\ell,R)\Big) = f(x)$$

  with probability 1 over the randomness R.

- **(Information-Theoretic Security)** For every function f and input y, the set of labels corresponding to f and y reveal nothing other than $f(y)$. Specifically, there exists a randomized simulator $\mathsf{DRE.Sim}$ that depends only on the output $f(x)$ such that

$$\forall\, f \in \mathcal{F},\ x \in \{0,1\}^\ell \quad \left\{\tilde{F}_1(f, x_1, R), \ldots, \tilde{F}_\ell(f, x_\ell, R)\right\} \sim \mathsf{DRE.Sim}(f(x))$$

  where $\sim$ denotes that the two random variables are identically distributed.

- The *length* of the randomized encoding is the maximum length of $\tilde{F}(f, R)$ over all choices of $f \in \mathcal{F}$ and the randomness R.

We will utilize the fact that functions computable in low depth have small decomposable randomized encodings.

Theorem 4.1 ([Bar86, Kil88]). *If $\mathcal{F}$ is a family of functions such that a universal function for $\mathcal{F}$ $U(f, x) = f(x)$ for $\mathcal{F}$ can be computed by Boolean formulae of depth d (over the basis $\{\wedge, \vee, \neg\}$), then $\mathcal{F}$ has an information-theoretic decomposable randomized encoding of length $O(4^d)$.*

4.2 Private Key Functional Encryption

Let $\mathcal{F} = \left\{f : \{0,1\}^\ell \to \{0,1\}^k\right\}$ be a family of functions. A *private key functional encryption scheme for* $\mathcal{F}$ is a tuple of polynomial-time algorithms $\Pi_{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ with the following syntax and properties:

- $\mathsf{FE.Setup}$ takes a security parameter 1^κ and outputs a master secret key $\mathsf{FE.msk}$.

- $\mathsf{FE.KeyGen}$ takes a master secret key $\mathsf{FE.msk}$ and a function $f \in \mathcal{F}$ and outputs a secret key $\mathsf{FE.sk}_f$ corresponding to the function f.

- $\mathsf{FE.Enc}$ takes the master secret key $\mathsf{FE.msk}$ and an input $x \in \{0,1\}^\ell$ and outputs a ciphertext c corresponding to the input x.

- **(Correctness)** $\mathsf{FE.Dec}$ takes a secret key $\mathsf{FE.sk}$ corresponding to a function f and a ciphertext c corresponding to an input x and outputs $f(x)$. Specifically, for every $\mathsf{FE.msk}$ is in the support of $\mathsf{FE.Setup}$

$$\mathsf{FE.Dec}(\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f), \mathsf{FE.Enc}(\mathsf{FE.msk}, x)) = f(x)$$

- The *key length* is the maximum length of $\mathsf{FE.sk}$ over all choices of $f \in \mathcal{F}$ and the randomness of $\mathsf{FE.Setup}, \mathsf{FE.Enc}$. The *ciphertext length* is the maximum length of c over all choices of $x \in \{0,1\}^\ell$ and the randomness of $\mathsf{FE.Setup}, \mathsf{FE.Enc}$.

- **(Security)** We will use a *non-adaptive* simulation-based definition of security. In particular, we are interested in security for a large number of keys n and a small number of ciphertexts m. We define security through the pair of games in Figure 3. We say that Π_{FE} is (n, m, ε)-secure if there exists a polynomial-time simulator $\mathsf{FE.Sim}$ such that for every polynomial-time adversary $\mathcal{A}$ and every κ,

$$\left|\mathbb{P}\left[E^{\mathrm{real}}_{\kappa,n,m}(\Pi_{FE}, \mathcal{A}) = 1\right] - \mathbb{P}\left[E^{\mathrm{ideal}}_{\kappa,n,m}(\Pi_{FE}, \mathcal{A}, \mathsf{FE.Sim}) = 1\right]\right| \le \varepsilon(\kappa)$$

Our goal is to construct a functional encryption scheme that is $(n, 2, \frac{1}{300n^3})$-secure and has short ciphertexts and keys, where $n = n(\kappa)$ is a polynomial in the security parameter. Although it is not difficult to see, in Section 7 we prove that the definition of security above implies the definition of two-index-hiding security that we use in Lemma 3.4.

$E^{\text{real}}_{\kappa,n,m}(\Pi_{FE}, \mathcal{A})$:
$\mathcal{A}$ outputs at most n functions $f_1, \dots, f_n$ and m inputs $x_1, \dots, x_m$
Let $\mathsf{FE.msk} \leftarrow_{\text{R}} \mathsf{FE.Setup}(1^\kappa)$ and

$$\forall\ i \in [n] \quad \mathsf{FE.sk}_{f_i} \leftarrow_{\text{R}} \mathsf{FE.KeyGen}(\mathsf{FE.msk}, f_i)$$
$$\forall\ j \in [m] \quad c_j \leftarrow_{\text{R}} \mathsf{FE.Enc}(\mathsf{FE.msk}, x_j)$$

$\mathcal{A}$ receives $\{\mathsf{FE.sk}_{f_i}\}_{i=1}^n$ and $\{c_j\}_{j=1}^m$ and outputs b

$E^{\text{ideal}}_{\kappa,n,m}(\Pi_{FE}, \mathcal{A}, \mathsf{FE.Sim})$:
$\mathcal{A}$ outputs at most n functions $f_1, \dots, f_n$ and m inputs $x_1, \dots, x_m$

$$\left(\left\{\mathsf{FE.sk}_{f_i}\right\}_{i=1}^n, \left\{c_j\right\}_{j=1}^m\right) \leftarrow_{\text{R}} \mathsf{FE.Sim}\left(\left\{f_i, \left\{f_i(x_j)\right\}_{j=1}^m\right\}_{i=1}^n\right)$$

$\mathcal{A}$ receives $\{\mathsf{FE.sk}_{f_i}\}_{i=1}^n$ and $\{c_j\}_{j=1}^m$ and outputs b

Figure 3: Security of Functional Encryption

4.2.1 Function-Hiding Functional Encryption

As an ingredient in our construction we also need a notion of **function-hiding security** for a (one-message) functional encryption scheme. Since we will only need this definition for a single message, we will specialize to that case in order to simplify notation. We say that Π_{FE} is function-hiding $(n, 1, \varepsilon)$-secure if there exists a polynomial-time simulator $\mathsf{FE.Sim}$ such that for every polynomial-time adversary $\mathcal{A}$ and every κ,

$$\left|\mathbb{P}\left[\bar{E}^{\text{real}}_{\kappa,n,1}(\Pi_{FE}, \mathcal{A}) = 1\right] - \mathbb{P}\left[\bar{E}^{\text{ideal}}_{\kappa,n,1}(\Pi_{FE}, \mathcal{A}, \mathsf{FE.Sim}) = 1\right]\right| \le \varepsilon(\kappa)$$

where $\bar{E}^{\text{real}}_{\kappa,n,1}, \bar{E}^{\text{ideal}}_{\kappa,n,1}$ are the same experiments as as $E^{\text{real}}_{\kappa,n,1}, E^{\text{ideal}}_{\kappa,n,1}$ except that the simulator in $\bar{E}^{\text{ideal}}_{\kappa,n,1}$ is not given the functions f_i as input. Namely, in $\bar{E}^{\text{ideal}}_{\kappa,n,1}$:

$$\left(\left\{\mathsf{FE.sk}_{f_i}\right\}_{i=1}^n, c\right) \leftarrow_{\text{R}} \mathsf{FE.Sim}\left(\{f_i(x)\}_{i=1}^n\right)$$

A main ingredient in the construction will be a function-hiding functional encryption scheme that is $(n, 1, \text{negl}(\kappa))$-secure. The construction is a small variant of the constructions of Sahai and Seyalioglu [SS10] and Gorbunov, Vaikuntanathan, and Wee [GVW12]

Theorem 4.2 (Variant of [SS10, GVW12])**.** *Let $\mathcal{F}$ be a family of functions such that a universal function for $\mathcal{F}$ has a decomposable randomized encoding of length L. That is, the function $U(f, x) = f(x)$ has a DRE of length L. If one-way functions exist, then for any polynomial $n = n(\kappa)$ there is an $(n, 1, \text{negl}(\kappa))$-function-hiding-secure functional encryption scheme Π with key length L and ciphertext length $O(\kappa L)$.*

Although this theorem follows in a relatively straightforward way from the techniques of [SS10, GVW12], we will give a proof of this theorem in Section 5. The main novelty in the theorem is to verify that in settings where we have a very short DRE—shorter than the security parameter κ—we can make the secret keys have length proportional to the length of the DRE rather than proportional to the security parameter.

5 One-Message Functional Encryption

We will now construct Π_{OFE} = (OFE.Setup, OFE.KeyGen, OFE.Enc, OFE.Dec): a function-hiding $(n, 1, \mathrm{negl}(\kappa))$-secure functional encryption scheme for functions with an (information-theoretic) decomposable randomized encoding DRE. The construction is essentially the same as the (public key) variants given by [SS10, GVW12] except we consider information theoretic randomized encodings instead of computationally secure ones and instead of encrypting the labels under a public key encryption scheme, we take advantage of the private-key setting to use an encryption method that produces ciphertexts with size equal to the message if the message is smaller than the security parameter. Encrypting the labels of a short randomized encoding, this allows us to argue that keys for our scheme are small. To perform this encryption, we use a PRF evaluated on known indices to mask each short label of DRE.

Let $n = \mathrm{poly}(\kappa)$ denote the number of users for the scheme. We assume for simplicity that $\lg n$ is an integer. Our construction will rely on the following primitives:

- A pseudorandom function family $\{\mathsf{PRF}_{\mathsf{sk}} : \{0,1\}^{\lg n} \to \{0,1\}^{\lg n} \mid s \in \{0,1\}^\kappa\}$.
- A decomposable randomized encoding of $f_y(x) = \mathbb{I}\{x \geq y\}$ where $x, y \in \{0,1\}^{\log n}$.

```
Setup(1^κ):
    Choose seeds $\mathsf{sk}_{k,b} \leftarrow_R \{0,1\}^\kappa$ for $k \in [\lg n], b \in \{0,1\}$.
    Choose randomness $R_j$ for the randomized encoding for each $j \in [n]$.
    Choose $x \leftarrow_R \{0,1\}^{\lg n}$.
    Define
        $\left\{K_{k,b}^{(j)}\right\} := \left\{\mathsf{PRF}_{\mathsf{sk}_{k,b}}(j) \oplus \tilde{F}_k(f_j, x_k \oplus b, R_j)\right\}$
    where $k \in \{1, .., \lg n\}, b \in \{0,1\}$
    Let each user's secret key be $sk_j = (j, \{K_{k,b}^{(j)}\})$.
    Let the master key be $\mathsf{msk} = \{\mathsf{sk}_{k,b}\}$.

Enc($i, \mathsf{msk} = \{\mathsf{sk}_{k,b}\}$):
    Output $c_i = (y := x \oplus i, \mathsf{sk}_{1,y_1}, \ldots, \mathsf{sk}_{\lg n, y_{\lg n}})$

Dec($c_i, sk_j$):
    Output $\mathsf{DRE.Decode}(\mathsf{PRF}_{\mathsf{sk}_{1,y_1}}(j) \oplus K_{1,y_1}^{(j)}, \ldots, \mathsf{PRF}_{\mathsf{sk}_{\lg n, y_{\lg n}}}(j) \oplus K_{\lg n, y_{\lg n}}^{(j)})$
```

Figure 4: Our scheme Π_{OFE}.

5.1 Proof of Correctness

$$\begin{aligned}
\mathsf{Dec}(c_i, \mathsf{sk}_j) &= \mathsf{DRE.Decode}(\mathsf{PRF}_{\mathsf{sk}_{1,y_1}}(j) \oplus K_{1,y_1}^{(j)}, \ldots, \mathsf{PRF}_{\mathsf{sk}_{\lg n, y_{\lg n}}}(j) \oplus K_{\lg n, y_{\lg n}}^{(j)}) \\
&= \mathsf{DRE.Decode}(\tilde{F}_1(f_j, y_1 \oplus x_1, R_j), \ldots, \tilde{F}_{\lg n}(f_j, y_{\lg n} \oplus x_{\lg n}, R_j)) \\
&= \mathsf{DRE.Decode}(\tilde{F}_1(f_j, i_1, R_j), \ldots, \tilde{F}_{\lg n}(f_j, i_{\lg n}, R_j)) \\
&= f_j(i)
\end{aligned}$$

Where the last step uses the (perfect) correctness of the randomized encoding scheme. So:

$$\mathop{\mathbb{P}}_{\vec{\mathsf{sk}}=\mathsf{Setup}(\kappa), c_i=\mathsf{Enc}(\mathsf{msk},i)}[\mathsf{Dec}(c_i, \mathsf{sk}_i) = \mathbb{I}\{i \le j\}] = \mathop{\mathbb{P}}_{R}\left[\mathsf{DRE.Decode}(\tilde{F}_1(f_j, i_1, R_j), \ldots, \tilde{F}_{\lg n}(f_j, i_{\lg n}, R_j)) = f_j(i)\right] = 1.$$

5.2 Proof of Security

Lemma 5.1. $\left|\mathbb{P}\left[E^{\text{real}}_{\kappa,n,m}(\Pi_{OFE}, \mathcal{A}) = 1\right] - \mathbb{P}\left[E^{\text{ideal}}_{\kappa,n,m}(\Pi_{OFE}, \mathcal{A}, \mathsf{FE.Sim}) = 1\right]\right| \le \varepsilon(\kappa)$

Proof. Consider the hybrid scheme Π^*_{OFE} defined in Figure 5, which uses a truly random string instead of the output of a PRF for the encrypted labels corresponding to the off-bits of $y = x \oplus i$. Note that this scheme is only useful in the nonadaptive security game, where i is known at time of Setup (since it is needed to compute y). We can easily show that the scheme is indistinguishable from the original scheme in the nonadaptive security game.

```
Setup(1^κ) :
  Choose seeds $\mathsf{sk}_k \leftarrow_{\text{R}} \{0,1\}^\kappa$ for $k \in [\lg n]$.
  Choose randomness $R_j$ for the randomized encoding for each $j \in [n]$.
  Choose $x \leftarrow_{\text{R}} \{0,1\}^{\lg n}$.
  Let $y = x \oplus i$.
  Define
      $\left\{K^{(j)}_{k,y_k}\right\} := \left\{\mathsf{PRF}_{\mathsf{sk}_k}(j) \oplus \tilde{F}_k(f_j, i_k, R_j)\right\}$
  Construct
      $\left\{K^{(j)}_{k,\overline{y}_k}\right\} \leftarrow_{\text{R}} \{0,1\}^{\lg n}$
  where $k \in \{1,..,\lg n\}$
  Let each user's secret key be $sk_j = (j, \{K^{(j)}_{k,b}\})$.
  Let the master key be $\mathsf{msk} = \{\mathsf{sk}_k\}$.

Enc($i$, msk = {sk_k}) :
  Output $c_i = (y = x \oplus i, \mathsf{sk}_1, \ldots, \mathsf{sk}_{\lg n})$

Dec($c_i, sk_j$):
  Output $\mathsf{DRE.Decode}(\mathsf{PRF}_{\mathsf{sk}_1}(j) \oplus K^{(j)}_{1,y_1}, \ldots, \mathsf{PRF}_{\mathsf{sk}_{\lg n}}(j) \oplus K^{(j)}_{\lg n, y_{\lg n}})$
```

Figure 5: Hybrid scheme Π^*_{OFE}.

Lemma 5.2. $\left|\mathbb{P}\left[E^{\text{real}}_{\kappa,n,m}(\Pi_{OFE}, \mathcal{A}) = 1\right] - \mathbb{P}\left[E^{\text{real}}_{\kappa,n,m}(\Pi^*_{OFE}, \mathcal{A}) = 1\right]\right| \le \lg n \cdot \mathsf{PRF.Adv}(\kappa) = \varepsilon(\kappa)$

Proof. Follows easily by the security of the PRF (applied $\lg n$ times in a hybrid for each function in the off-bits of y). □

In Figure 6 we define a simulator for the ideal setting that is indistinguishable from our hybrid scheme Π^*_{OFE}. The simulator uses the simulator for the decomposable randomized encoding scheme to generate the labels to be encrypted using only the knowledge of the output value of the functions on the input.

```
OFE.Sim:
Input: $1^\kappa$, and the evaluation of $n$ (unknown) functions on 1 unknown input: $\{y_i\}_{i=1}^n$
Let DRE.Sim be the simulator for information-theoretic randomized encoding DRE.

// Generate ciphertext
Choose $y \leftarrow_R \{0,1\}^{\lg n}$.
For $k \in [\lg n]$:
  Choose $\mathsf{sk}_k \leftarrow_R \{0,1\}^\kappa$
Let $c = (y, \mathsf{sk}_1, \ldots, \mathsf{sk}_{\lg n})$

// Generate keys for $j \in [n]$ using DRE.Sim
For $j \in [n]$:
  Generate: $\left(\tilde{F}_1^{(j)}, \ldots, \tilde{F}_{\lg n}^{(j)}\right) \leftarrow \mathsf{DRE.Sim}(y_j)$
  Let: $K_{k,y_k}^{(j)} = \mathsf{PRF}_{\mathsf{sk}_k}(j) \oplus \tilde{F}_k^{(j)}$ for $k \in [\lg n]$
  Choose: $K_{k,\overline{y}_k}^{(j)} \leftarrow_R \{0,1\}^{\lg n}$ for $k \in [\lg n]$
  Let: $\mathsf{OFE.sk}_j = \left\{K_{k,b}^{(j)}\right\}_{k \in [\lg n], b \in \{0,1\}}$

// Output the $n$ simulated secret keys and simulated ciphertext
Output: $\{\mathsf{OFE.sk}_i\}_{i=1}^n$, $c$
```

Figure 6: The Simulator OFE.Sim for Π_{OFE}.

Lemma 5.3. $\left|\mathbb{P}\left[E_{\kappa,n,m}^{\text{real}}(\Pi_{OFE}^*, \mathcal{A}) = 1\right] - \mathbb{P}\left[E_{\kappa,n,m}^{\text{ideal}}(\Pi_{OFE}, \mathcal{A}, \mathsf{FE.Sim}) = 1\right]\right| = 0$

Proof. Follows easily by the information-theoretic security of the randomized encoding. □

Adding the statements of Lemma 5.2 and Lemma 5.3 gives us the original statement of security. □

So, Π_{OFE} is a function-hiding-secure private-key functional encryption scheme $(n, 1, \text{negl}(\kappa))$ with security based on the hardness of a PRF (which can be instantiated using a one-way function) and the existence of an information-theoretic randomized encoding for the family of comparison functions $\{f_j : \{0,1\}^{\lg n} \to \{0,1\}\}$ of length L. Furthermore, note that the length of ciphertexts is: $\lg n \cdot \kappa + \lg n = O(\kappa L)$ and the length of each key is L, satisfying the conditions of Theorem 4.2.

6 A Two-Message Functional Encryption Scheme for Comparison

We now use the one-message functional encryption scheme Π_{OFE} described in Section 5 to construct a functional encryption scheme Π_{FE} that is $(n, 2, \frac{1}{300n^3})$-secure for the family of comparison functions. For any $y \in \{0,1\}^\ell$, let

$$f_y(x) = \mathbb{I}\{x \geq y\}$$

where the comparison operation treats x, y as numbers in binary. We define the family of functions

$$\mathcal{F}_{\text{comp}} := \left\{f_y : \{0,1\}^\ell \to \{0,1\} \;\middle|\; y \in \{0,1\}^\ell\right\}$$

In our application, we need $x, y \in \{0, 1, \ldots, n\}$, so we will set $\ell = \lceil \log_2(n+1) \rceil = O(\log n)$. One important property of our construction is that the user key length will be fairly small as a function of ℓ, so that when $\ell = O(\log n)$, the overall length of user keys will be $n^{o(1)}$ (in fact, nearly polylogarithmic in n).

6.1 Construction

Our construction will be for a generic family of functions $\mathcal{F}$, and we will only specialize the construction to $\mathcal{F}_{\text{comp}}$ when setting the parameters and bounding the length of the scheme. Before giving the formal construction, let's gather some notation and ingredients. Note that we will introduce some additional parameters that are necessary to specify the scheme, but we will leave many of these parameters to be determined later.

- Let n be a parameter bounding the number of user keys in the scheme, and let $\mathbb{F}$ be a finite field whose size we will determine later.
- Let $\mathcal{F} = \left\{ f : \{0,1\}^\ell \to \{0,1\} \right\}$ be a family of functions. For each function $f \in \mathcal{F}$ we define an associated polynomial $\tilde{f} : \mathbb{F}^{\ell+1} \to \mathbb{F}$ as follows:
  1. Let $\hat{f} : \mathbb{F}^\ell \to \mathbb{F}$ be a polynomial computing f
  2. Define $\tilde{f} : \mathbb{F}^{\ell+1} \to \mathbb{F}$ to be $\tilde{f}(x_1, \ldots, x_\ell, z) = \hat{f}(x_1, \ldots, x_\ell) + z$

  Let D and S be such that every for every $f \in \mathcal{F}$, the associated polynomial $\tilde{f}$ has degree at most D and can be computed by an arithmetic circuit of size at most S. These degree and size parameters will depend on $\mathcal{F}$.
- Let $\mathcal{P}_{D',S',\mathbb{F}}$ be the set of all univariate polynomials $p : \mathbb{F} \to \mathbb{F}$ of degree at most D' and size at most S'. Let $\Pi_{OFE} = (\mathsf{OFE.Setup}, \mathsf{OFE.KeyGen}, \mathsf{OFE.Enc}, \mathsf{OFE.Dec})$ be an $(n, 1, \mathsf{negl}(\kappa))$-function-hiding-secure functional encryption scheme (i.e. secure for n keys and one message) for the family of polynomials $\mathcal{P}_{D',S',\mathbb{F}}$.

We're now ready to describe the construction of the two-message functional encryption scheme Π_{FE}. The scheme is specified in Figure 7.

Correctness of Π_{FE}. Before going on to prove security, we will verify that encryption and decryption are correct for our scheme. Fix any $f_i \in \mathcal{F}$ and let $\tilde{f}_i : \mathbb{F}^\ell \to \mathbb{F}$ be the associated polynomial, and fix any input $x \in \mathbb{F}^\ell$. Let $r_i : \mathbb{F} \to \mathbb{F}$ be the degree RD polynomial chosen by $\mathsf{FE.KeyGen}$ on input f_i and let $\tilde{f}_{i,r_i}(\cdot, t)$ be the function used to generate the key $\mathsf{OFE.sk}_{i,t}$. Let $q : \mathbb{F} \to \mathbb{F}^\ell$ be the degree R polynomial map chosen by $\mathsf{FE.Enc}$ on input x. Observe that, by correctness of Π_{OFE}, when we run $\mathsf{FE.Dec}$ we will have

$$\tilde{p}(t) = \mathsf{OFE.Dec}(\mathsf{OFE.sk}_{i,t}, c_t) = \tilde{f}_{i,r_i}(q(t), t) = \tilde{f}_i(q(t)) + r_i(t).$$

Now, consider the polynomial $\tilde{f}_{i,q,r_i} : \mathbb{F} \to \mathbb{F}$ defined by

$$\tilde{f}_{i,q,r_i}(t) = \tilde{f}_i(q(t)) + r_i(t).$$

Since $\tilde{f}_i$ has degree at most D, q has degree at most R, and r_i has degree at most RD, the degree of $\tilde{f}_{i,q,r_i}$ is at most RD. Since $|\mathcal{U}| = RD + 1$, the polynomial $\tilde{p}$ agrees with $\tilde{f}_{i,q,r_i}$ at $RD + 1$ distinct points, and thus $\tilde{p} \equiv \tilde{f}_{i,q,r_i}$. In particular, $\tilde{p}(0) = \tilde{f}_{i,q,r_i}(0)$. Since we chose r_i and q such that $r_i(0) = 0$ and $q(0) = x$, we have $\tilde{p}(0) = \tilde{f}_i(q(0)) + r_i(0) = \tilde{f}_i(x)$. This completes the proof of correctness.

```
Global Parameters: A family of functions $\mathcal{F} = \{f : \{0,1\}^\ell \to \{0,1\}^\ell\}$ is a family of functions, a finite field $\mathbb{F}$, a bound $D$ on the degree of polynomials $\tilde{f} : \mathbb{F}^{\ell+1} \to \mathbb{F}$ computing a function in $\mathcal{F}$, a parameter $R \in \mathbb{N}$, and parameters $U = RD + 1$, and $T = U^2$.

$\mathsf{FE.Setup}(1^\kappa)$ :
  Generate $T$ independent master keys for $\mathsf{OFE.msk}_t \leftarrow_R \mathsf{OFE.Setup}(1^\kappa)$
  Output $\mathsf{FE.msk} = \{\mathsf{OFE.msk}_t\}_{t=1}^T$

$\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f_i)$ :
  // To aid in our proofs later, we index invocations of FE.KeyGen with $i$
  Let $\tilde{f}_i : \mathbb{F}^\ell \to \mathbb{F}$ be a multivariate polynomial computing $f_i$
  Choose a random polynomial $r_i : \mathbb{F} \to \mathbb{F}$ of degree $RD$ such that $r_i(0) = 0$
  For every $t \in [T]$:
    Define the function $\tilde{f}_{i,r_i}(x,t) = \tilde{f}_i(x) + r_i(t)$
    Let $\mathsf{OFE.sk}_{i,t} \leftarrow_R \mathsf{OFE.KeyGen}(\mathsf{OFE.msk}_t, \tilde{f}_{i,r_i}(\cdot, t))$
  Output $\mathsf{FE.sk}_i = \{\mathsf{OFE.sk}_{i,t}\}_{t=1}^T$

$\mathsf{FE.Enc}(\mathsf{FE.msk}, x)$ :
  Choose a random polynomial map $q : \mathbb{F} \to \mathbb{F}^\ell$ of degree $R$ so that $q(0) = x$.
  Choose a random $\mathcal{U} \subseteq [T]$ of size $U$
  For every $t \in \mathcal{U}$: let $c_t \leftarrow_R \mathsf{OFE.Enc}(\mathsf{OFE.msk}_t, q(t))$
  Output $c = \{c_t\}_{t \in \mathcal{U}}$

$\mathsf{FE.Dec}(\mathsf{FE.sk}_i, c)$ :
  Let $\mathsf{FE.sk}_i = \{\mathsf{OFE.sk}_{i,t}\}_{t=1}^T$, $c = \{c_t\}_{t \in \mathcal{U}}$
  For every $t \in \mathcal{U}$, let $\tilde{p}(t) = \mathsf{OFE.Dec}(\mathsf{OFE.sk}_{i,t}, c_t)$
  Extend the polynomial $\tilde{p}(t)$ to all of $\mathbb{F}$ by interpolation
  Output $\tilde{p}(0)$
```

Figure 7: A Functional Encryption Scheme for 2 Messages

6.2 Security for Two Messages

Theorem 6.1. *For every polynomial $n = n(\kappa)$, Π_{FE} is $(n, 2, \delta)$-secure for $\delta = T \cdot \mathrm{negl}(\kappa) + 2^{-\Omega(R)}$.*

First we describe at a high level how to simulate. To do so, it will be useful to first introduce some terminology. Recall that $\mathsf{FE.Setup}$ instantiates T independent copies of the one-message scheme T. We refer to each instantiation as a *component*. Thus, when we talk about generating a secret key for a function f_i we will talk about generating each of the T components of that key and similarly when we talk about generating a ciphertext for an input x_b we will talk about generating each of the U components of that ciphertext. Thus the simulator has to generate a total of nT components of keys and $2U$ components of ciphertexts. The simulator will consider several types of components:

- Components $t \in \mathcal{U}_1 \cap \mathcal{U}_2$ where $\mathcal{U}_1, \mathcal{U}_2$ are the random sets of components chosen by the encryption scheme for the two inputs, respectively. The adversary obtains two ciphertexts for these components, so we cannot use the simulator for the one-message scheme. Thus for these components we simply choose uniformly random values for all the keys and ciphertexts and use the real one-message scheme.

- Components $t \in \mathcal{U}_1 \triangle \mathcal{U}_2$ (where $\triangle$ is the symmetric difference). For these we want to use the simulator for the one-message scheme to generate both the keys for each function and the ciphertexts for these components (recall that the one-message scheme is function-hiding). To do so, we need to feed the simulator with the evaluation of each of the functions on the chosen input. We show how to generate these outputs by leveraging the random high-degree polynomials r_i included in the keys. These values are then fed into the simulator to produce the appropriate key and ciphertext components.

- Components $t \notin \mathcal{U}_1 \cup \mathcal{U}_2$. For these components the real scheme would not generate a ciphertext so the distribution can be simulated by a simulator that takes no inputs.

With this outline in the place, it is not too difficult to construct and analyze the simulator.

Proof of Theorem 6.1. We prove security via the simulator described in Figure 8.

First we make a simple claim showing that there is only a small probability that the simulator has to halt and output $\perp$ because $\mathcal{I}$ is too large.

Claim 6.2. $\mathbb{P}[\mathsf{FE.Sim} = \perp] = 2^{-\Omega(R)}$.

Proof Sketch for Claim 6.2. Recall that $\mathcal{I}$ is defined to be $\mathcal{U}_1 \cap \mathcal{U}_2$. Since $\mathcal{U}_1, \mathcal{U}_2$ are random subsets of $[T]$, each of size U, and we set $T = U^2$, we have $\mathbb{E}[|\mathcal{I}|] = 1$. Moreover, the intersection of the two sets has a hypergeometric distribution, and by a standard tail bound for the hypergeometric distribution we have $\mathbb{P}[\mathsf{FE.Sim} = \perp] = \mathbb{P}[|\mathcal{I}| > R] \leq 2^{-\Omega(R)}$. □

In light of the above claim, we will assume for the remainder of the analysis that the simulator does not output $\perp$, and thus $|\mathcal{I}| \leq R$, and this will only cost add $2^{-\Omega(R)}$ to the simulation error. In what follows, we will simplify notation by referring only to components corresponding to keys and one of the ciphertexts, and will drop the superscript b. All of our arguments also applies to the second ciphertext, since this ciphertext is generated in a completely symmetric way.

Components Used in Both Ciphertexts. First, we claim that the simulator produces the correct distribution of the keys and ciphertexts for the components $t \in \mathcal{I}$. Note that the simulator chooses the keys in exactly the same way as the real scheme would: it generates keys for the functions $\tilde{f}_{i,r_i}(\cdot, t)$ where r_i is a random degree RD polynomial with the constant coefficient 0. The ciphertexts in the real scheme would contain the messages $\{q(t)\}_{t \in \mathcal{U}}$ where q is a random degree R polynomial with constant coefficient equal to the (unknown) input x. Since $|\mathcal{I}| \leq R$, this is a uniformly random set of values. Thus, the distribution of $\{\alpha_t\}_{t \in \mathcal{U}}$ is identical to $\{q(t)\}_{t \in \mathcal{U}}$, and therefore the simulated ciphertext components and the real ciphertext components have the same distribution.

Components Used in Exactly One Ciphertext. Next we claim that the simulated keys and ciphertexts for the components $t \in \mathcal{U} \setminus \mathcal{I}$ are computationally indistinguishable from those of the real scheme. Since in these components we only need to generate a single ciphertext, we can rely on the simulator $\mathsf{OFE.Sim}$ for the one-message scheme. $\mathsf{OFE.Sim}$ takes evaluations of n functions each at a single input and simulates the keys for those n functions and the ciphertext for that single input. In order to apply the indistinguishability guarantee for $\mathsf{OFE.Sim}$, we need to argue that the evaluations that $\mathsf{FE.Sim}$ feeds to $\mathsf{OFE.Sim}$ are jointly identically distributed to the real scheme.

Recall that in the real scheme, each key corresponds to a function $\tilde{f}_{i,r_i}(\cdot, t)$ and this function gets evaluated on points $q(t)$. Thus for each function i, and each ciphertext component t, the evaluation is $\tilde{f}_{i,q,r_i}(t) = \tilde{f}_i(q(t)) + r_i(t)$. The polynomials $q, r_1, \ldots, r_n$ are chosen so that for every i, $\tilde{f}_{i,q,r_i}(0) = \tilde{f}_i(x)$ where x is the (unknown) input. We need to argue that the set of evaluations $\{\tilde{y}_{i,t}\}$ generated by the

```
FE.Sim:
Input: $1^\kappa$, $n$ functions and their evaluations on 2 unknown inputs $\left\{f_i, y_i^1, y_i^2\right\}_{i=1}^n$
Let $\tilde{f}_i : \mathbb{F}^{\ell+1} \to \mathbb{F}$ and $\tilde{y}_i^1, \tilde{y}_i^2 \in \mathbb{F}$ be the associated polynomial and field elements
Let OFE.Sim be the simulator for the one-message scheme $\Pi_{OFE}$

Choose random $r_1, \ldots, r_n : \mathbb{F} \to \mathbb{F}$ of degree $RD$ s.t. for all $i$, $r_i(0) = 0$, let $\tilde{f}_{i,r_i}(\cdot, t) = \tilde{f}_i(\cdot) + r_i(t)$
Choose two random sets $\mathcal{U}^1, \mathcal{U}^2 \subseteq [T]$ of size $U = RD + 1$
Let $\mathcal{I} = \mathcal{U}^1 \cap \mathcal{U}^2$. If $|\mathcal{I}| > R$, halt and output $\perp$. Assume from here on that $|\mathcal{I}| \leq R$.

// Generate the keys and ciphertexts for components $t \in \mathcal{I}$ using $\Pi_{OFE}$
For $t \in \mathcal{I}$:
  For $b \in \{1, 2\}$ choose a random value $\alpha_t^b \in \mathbb{F}^\ell$
  Let $\mathsf{OFE.msk}_t \leftarrow_R \mathsf{OFE.Setup}(1^\kappa)$
  For every $i$, let $\mathsf{OFE.sk}_{i,t} \leftarrow_R \mathsf{OFE.KeyGen}(\mathsf{OFE.msk}_t, \tilde{f}_{i,r_i}(\cdot, t))$
  For $b \in \{1, 2\}$, let $c_t^b \leftarrow_R \mathsf{OFE.Enc}(\mathsf{OFE.msk}_t, \alpha_t^b)$

// Interpolate consistent evaluations to give OFE.Sim for $t \in \mathcal{U}^b \setminus \mathcal{I}$
For every $i$, generate a set of evaluations $\left\{\tilde{y}_{i,t}^b\right\}_{t \in \mathcal{U}^b}$ as follows:
  // Choose random values consistent with the choices we made for $t \in \mathcal{I}$
  For every $i$ and every $t \in \mathcal{I}$, set $\tilde{y}_{i,t}^b = \tilde{f}_{i,r_i}(\alpha_t^b)$
  For all except one point $t \in \mathcal{U}^b \setminus \mathcal{I}$, choose uniformly random values for $\tilde{y}_{i,t}^b$
  // Ensure consistency with the final output.
  For all $i$, interpolate a polynomial $\tilde{p}_i^b$ of degree $RD$ such that $\tilde{p}_i^b(0) = \tilde{y}_i^b$, $\tilde{p}_i^b(t) = \tilde{y}_{i,t}^b$
  For the last point $t \in \mathcal{U}^b$, set $\tilde{y}_{i,t}^b = \tilde{p}_i^b(t)$

// Generate keys and ciphertexts for $t \in \mathcal{U}_b \setminus \mathcal{I}$ using OFE.Sim
For $t \in \mathcal{U}_b \setminus \mathcal{I}$: $\left(\{\mathsf{OFE.sk}_{i,t}\}_{i=1}^n, c_t\right) \leftarrow_R \mathsf{OFE.Sim}\left(\left\{\tilde{y}_{i,t}^b\right\}_{i=1}^n\right)$

// Generate keys (but no ciphertexts) for $t \notin \mathcal{U}_1 \cup \mathcal{U}_2$ obliviously
For $t \in [T] \setminus (\mathcal{U}^1 \cup \mathcal{U}^2)$ and $i \in [n]$, let $\mathsf{OFE.sk}_{i,t} \leftarrow_R \mathsf{OFE.Sim}()$.

// Output the $n$ simulated secret keys and 2 simulated ciphertexts
Output: $\left\{\{\mathsf{OFE.sk}_{i,t}\}_{t=1}^T\right\}_{i=1}^n, \left\{c_t^1\right\}_{t \in \mathcal{U}_1}, \left\{c_t^2\right\}_{t \in \mathcal{U}_2}$
```

Figure 8: The Simulator FE.Sim for Π_{FE}.

simulator have the same distribution as $\{\tilde{f}_{i,q,r_i}(t)\}$. Observe that, since r_i is a random polynomial of degree RD with constant coefficient 0, its evaluation on any set of RD points is jointly uniformly random. Therefore, for every q chosen independently of r, the evaluation of $\tilde{f}_{i,q,r_i}$ on any set of RD points is also jointly uniformly random. On the other hand, the evaluation of $\tilde{f}_{i,q,r_i}$ on any set of $RD+1$ points determines the whole function and thus determines $\tilde{f}_{i,q,r_i}(0)$, therefore conditioned on evaluations at any set of RD points, and the desired value of $\tilde{f}_{i,q,r_i}(0)$, the evaluation at any other point is uniquely determined.

Now, in the simulator, for every i, we choose RD evaluations $\tilde{y}_{i,t}$ uniformly randomly—for the points $t \in \mathcal{I}$ they are uniformly random because the polynomials r_i and the values $\alpha_{i,t}$ were chosen randomly, and then for all but one point in $\mathcal{U} \setminus \mathcal{I}$ we explicitly chose them to be uniformly random. For the remaining point, we chose $\tilde{y}_{i,t}$ to be the unique point such that we obtain the correct evaluation of $\tilde{f}_{i,q,r_i}(0)$, which is the value $\tilde{y}_i$ that was given to the simulator. Thus, we have argued that for any individual i, the distribution of the points $\tilde{y}_{i,t}$ that we give to the simulator is identical to that of the real scheme. The fact that this holds jointly over all i follows immediately by independence of the polynomials $r_1, \dots, r_n$.

Components Used in Neither Ciphertext. Since the underlying one-message scheme satisfies function-hiding, it must be the case that the distribution of n keys and no messages is computationally indistinguishable from a fixed distribution. That is, it can be simulated given no evaluations. Thus we can simply generate the keys for these unused components in a completely oblivious way.

Since we have argued that all components are simulated correctly, we can complete the proof by taking a hybrid argument over the simulation error for each of the T components, and a union bound over the failure probability corresponding to the case where $|\mathcal{I}| > R$. Thus we argue that FE.Sim and the real scheme are computationally indistinguishable with the claimed parameters. □

6.3 Bounding the Scheme Length for Comparison Functions

In the application to differential privacy, we need to instantiate the scheme for the family of comparison functions of the form $f_y(x) = \mathbb{I}\{x \geq y\}$ where $x, y \in \{0,1\}^{\log n}$, and we need to set the parameters to ensure $(n, 2, \frac{1}{300n^3})$-security where $n = n(\kappa)$ is an arbitrary polynomial.

Theorem 6.3. *For every polynomial $n = n(\kappa)$ there is a $(n, 2, \frac{1}{300n^3})$-secure functional encryption scheme for the family of comparison functions on $O(\log n)$ bits with keys are in K_κ and ciphertexts in C_κ where*

$$|K_\kappa| = 2^{2^{\mathrm{poly}(\log\log n)}} = 2^{n^{o(1)}} \quad \textit{and} \quad |C_\kappa| = 2^\kappa.$$

Theorem 1.1 follows by combining Theorem 6.3 with Theorem 3.3. Note that Theorem 6.3 constructs a different scheme for every polynomial $n = n(\kappa)$. However, we can obtain a single scheme that is secure for every polynomial $n(\kappa)$ by instantiating this construction for some $n'(\kappa) = \kappa^{\omega(1)}$.

Proof of Theorem 6.3. By Theorem 6.1, if the underlying one-message scheme Π_{OFE} is $(n, 1, \mathrm{negl}(\kappa))$-function-hiding secure, then the final scheme Π_{FE} will be $(n, 2, \delta)$-secure for $\delta = T \cdot \mathrm{negl}(\kappa) + 2^{-\Omega(R)}$. If we choose an appropriate $R = \Theta(\log n)$ then we will have $\delta = T \cdot \mathrm{negl}(\kappa) + \frac{1}{600n^3}$. As we will see, T will be a polynomial in n, so for sufficiently large values of κ, we will have $\delta \leq \frac{1}{300n^3}$. To complete the proof, we bound the length of the keys and ciphertexts:

The functions constructed in FE.KeyGen *have small DREs.* For the family of comparison functions on $\log n$ bits, there is a universal Boolean formula $u(x,y) : \{0,1\}^{\log n} \times \{0,1\}^{\log n} \to \{0,1\}$ of size $S = O(\log n)$ and depth $d = O(\log\log n)$ that computes $f_y(x)$. Thus, for any field $\mathbb{F}$, the polynomial $\tilde{u}(x,y) : \mathbb{F}^{\log n} \times \mathbb{F}^{\log n} \to \mathbb{F}$ is computable by an arithmetic circuit of size $S = O(\log n)$ and depth $d = O(\log\log n)$, and this polynomial computes $\tilde{f}_y(x)$. For any value $r \in \mathbb{F}$, the polynomial $\tilde{u}_r(x,y) = \tilde{u}(x,y) + r$ is also computable by an arithmetic circuit of size $S + 1 = O(\log n)$ with degree d. Note that this polynomial is a universal evaluation for the polynomials $\tilde{f}_{y,r}(\cdot, t) = \tilde{f}_y(\cdot) + r(t)$ created in FE.KeyGen.

To obtain a DRE, we can write $\tilde{u}_r(x,y)$ as a Boolean formula $u_{r,\mathbb{F}}(x,y) : \{0,1\}^{(\log n)(\log|\mathbb{F}|)} \times \{0,1\}^{(\log n)(\log|\mathbb{F}|)} \to \{0,1\}^{\log|\mathbb{F}|}$ with depth $d' = d \cdot \mathrm{depth}(\mathbb{F})$ and size $S' = S \cdot \mathrm{size}(\mathbb{F})$ where $\mathrm{depth}(\mathbb{F})$ and $\mathrm{size}(\mathbb{F})$ are the depth and size of Boolean formulae computing operations in the field $\mathbb{F}$, respectively. Later we will argue that it suffices to choose a field of size $\mathrm{poly}(\log n)$, and thus $d_\mathbb{F}, S_\mathbb{F} = \mathrm{poly}(\log\log n)$. Therefore these functions can be computed by formulae of depth $d' = \mathrm{poly}(\log\log n)$

and size $S' = \mathrm{poly}(\log n)$. Finally, by Theorem 4.1, the universal evaluator for this family has DREs of length $O(4^{d'}) = \exp(\mathrm{poly}(\log\log n))$.

The secret keys and ciphertexts for each component are small. Π_{FE} generates key and ciphertext components for up to T independent instantiations of Π_{OFE}. Each function for Π_{OFE} corresponds to a formula of the form $u_{r,\mathbb{F}}$ defined above. By Theorem 4.2, we can instantiate Π_{OFE} so that each key component has length $\exp(\mathrm{poly}(\log\log n))$ and each ciphertext component has length $\kappa \cdot \exp(\mathrm{poly}(\log\log n)) = \mathrm{poly}(\kappa)$, where the last inequality is because $n = \mathrm{poly}(\kappa)$.

The number of components T and the size of the field $\mathbb{F}$ is small. In Π_{FE} we take $T = U^2 = (RD+1)^2$ where $D \le 2^d$ is the degree of the polynomials computing the comparison function over $\mathbb{F}$. As we argued above, we can take $R = O(\log n)$ and $D = \mathrm{poly}(\log n)$. Therefore we have $T = \mathrm{poly}(\log n)$. We need to ensure that $|\mathbb{F}| \ge T+1$, since the security analysis relies on the fact that each component $t \in [T]$ corresponds to a different non-zero element of $\mathbb{F}$. Therefore, it suffices to have $|\mathbb{F}| = \mathrm{poly}(\log n)$. In particular, this justifies the calculations above involving the complexity of field operations.

Putting it together. By the above, each component of the secret keys has length $\exp(\mathrm{poly}(\log\log n))$ and there are $\mathrm{poly}(\log n)$ components, so the overall length of the keys for Π_{FE} is $\exp(\mathrm{poly}(\log\log n))$. Each component of the ciphertexts has length $\mathrm{poly}(\kappa)$ and there are $\mathrm{poly}(\log n) = \mathrm{poly}(\log\kappa))$ components, so the overall length of the ciphertexts for Π_{FE} is $\mathrm{poly}(\kappa)$. The theorem statement now follows by rescaling κ and converting the bound on the length of the keys and ciphertexts to a bound on their number. □

7 Two-Message Functional Encryption Implies Index Hiding

As discussed in subsection 3.2, Lemma 3.4 tells us that if we can show that any adversary's advantage in the TwoIndexHiding game is small, then the game's traitor-tracing scheme satisfies weak index-hiding security and gives us the lower bound of Theorem 3.3. First, note that one can use a private key functional encryption scheme for comparison functions directly as a traitor-tracing scheme, since they have the same functionality. We will now show that any private key functional encryption scheme that is $(n, 2, \frac{1}{300n^3})$-secure is a secure traitor-tracing scheme in the TwoIndexHiding game.

In Figure 9, we describe a variant of the TwoIndexHiding game from Figure 2 that uses the simulator $\mathsf{FE.Sim}$ for the functional encryption scheme $\Pi_{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for comparison functions $f_y(x) = \mathbb{I}\{x \ge y\}$ where $x, y \in \{0,1\}^{\log n}$ that is $(n, 2, \frac{1}{300n^3})$-secure. Note that the challenger can give the simulator inputs that are independent of the game's b_0, b_1 since for all indices $j \ne i^*$, the output values of the comparison function for j on both inputs $i^* - b_0, i^* - b_1$ are always identical: $\mathbb{I}\{j > i^*\}$ (for all $b_0, b_1 \in \{0,1\}$).

The challenger runs the simulator to produce:

$$\left(\left\{\mathsf{sk}_j\right\}_{j \ne i^* \in [n]}, \{c_0, c_1\}\right) \leftarrow_{\mathrm{R}} \mathsf{FE.Sim}\left(\left\{f_j, \{\mathbb{I}\{j > i^*\}, \mathbb{I}\{j > i^*\}\}\right\}_{j \ne i^* \in [n]}\right)$$

The adversary A is given keys sk_{-i^*} and outputs a decryption program S.
Choose $b_0 \leftarrow_{\mathrm{R}} \{0,1\}$ and $b_1 \leftarrow_{\mathrm{R}} \{0,1\}$ independently.
Let $b' = S(c_0, c_1)$.

Figure 9: SimTwoIndexHiding$[i^*]$

Defining:

$$\text{SimTwoAdv}[i^*] = \underset{\text{SimTwoIndexHiding}[i^*]}{\mathbb{P}}[b' = b_0 \oplus b_1] - \frac{1}{2}$$

We can then prove the following lemmas:

Lemma 7.1. *For all p.p.t. adversaries,* $\text{SimTwoAdv}[i^*] = 0$*.*

Proof. In SimTwoIndexHiding$[i^*]$, b_0, b_1 are chosen uniformly at random and independent of the adversary's view. Therefore, the probability that the adversary outputs $b' = b_0 \oplus b_1$ is exactly $\frac{1}{2}$, and so $\text{SimTwoAdv}[i^*] = \underset{\text{SimTwoIndexHiding}[i^*]}{\mathbb{P}}[b' = b_0 \oplus b_1] - \frac{1}{2} = 0$. □

Lemma 7.2. *For all p.p.t. adversaries,* $|\text{TwoAdv}[i^*] - \text{SimTwoAdv}[i^*]| \le \frac{1}{300n^3}$*.*

Proof. This follows easily from the simulation security of the 2-message FE scheme. □

We can now show that any adversary's advantage in the TwoIndexHiding game is small:

Lemma 7.3. *Given a Two-Message Functional Encryption scheme for comparison functions* $f_y(x) = \mathbb{I}\{x \ge y\}$ *where* $x, y \in \{0,1\}^{\log n}$ *that is* $(n, 2, \frac{1}{300n^3})$*-secure,*

$$\Pi_{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$$

then for all i^**,*

$$\text{TwoAdv}[i^*] \le \frac{1}{300n^3}$$

Proof. Adding the statements of Lemma 7.1 and Lemma 7.2 gives us the statement of the lemma:

$$\text{TwoAdv}[i^*] \le \frac{1}{300n^3}$$ □

Combining Lemma 7.3 with Lemma 3.4, the $(n, 2, \frac{1}{300n^3})$-secure Two-Message Functional Encryption scheme from Section 6 is therefore a $(n, \{K_\kappa, C_\kappa\})$-traitor tracing scheme with weak index-hiding security. From Theorem 6.3, we have that

$$|K_\kappa| = 2^{2^{\text{poly}(\log\log n)}} = 2^{n^{o(1)}} \quad \text{and} \quad |C_\kappa| = 2^\kappa.$$

which when combined with Theorem 3.3 gives us our main Theorem 1.1.

Acknowledgements

The authors are grateful to Salil Vadhan for many helpful discussions about this research.

The first and second authors are supported in part by the Defense Advanced Research Project Agency (DARPA) and Army Research Office (ARO) under Contract W911NF-15-C-0236, and NSF grants CNS-1445424 and CCF-1423306. Any opinions, findings and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the the Defense Advanced Research Projects Agency, Army Research Office, the National Science Foundation, or the U.S. Government. The first author is also supported by NSF grant CNS-1552932 and NSF Graduate Research Fellowship DGE-16-44869.

The third author is supported by NSF grant CCF-1718088.

The fourth author is supported by NSF grants CNS-1314722, CNS-1413964.

References

[Bar86] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *Proceedings of the 18th ACM Symposium on Theory of Computing (STOC)*, 1986.

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

[BLR13] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. *J. ACM*, 60(2):12, 2013.

[BNS13] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (RANDOM-APPROX)*. 2013.

[BNS$^+$16] Raef Bassily, Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. Algorithmic stability for adaptive data analysis. In *Proceedings of the 48th Annual ACM on Symposium on Theory of Computing, STOC*, 2016.

[BNSV15] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Differentially private release and learning of threshold functions. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2015.

[BS98] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, pages 464–473. IEEE, October 18–21 2014.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Eurocrypt*, 2006.

[BU17] Mitali Bafna and Jonathan Ullman. The price of selection in differential privacy. In *COLT 2017 - The 30th Annual Conference on Learning Theory*, 2017.

[BUV14] Mark Bun, Jonathan Ullman, and Salil P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. In *STOC*, pages 1–10. ACM, May 31 – June 3 2014.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Proceedings of CRYPTO 2014*, 2014.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, 1994.

[CTUW14] Karthekeyan Chandrasekaran, Justin Thaler, Jonathan Ullman, and Andrew Wan. Faster private release of marginals on small databases. In *Innovations in Theoretical Computer Science (ITCS)*, 2014.

[DFH$^+$15] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Preserving statistical validity in adaptive data analysis. In *STOC*. ACM, 2015.

[DLS14] Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Symposium on Theory of Computing, (STOC)*, 2014.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TCC)*, 2006.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Principles of Database Systems (PODS)*. ACM, 2003.

[DN04] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, 2004.

[DNR$^+$09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Symposium on Theory of Computing (STOC)*. ACM, 2009.

[DNT14] Cynthia Dwork, Aleksandar Nikolov, and Kunal Talwar. Using convex relaxations for efficiently and privately releasing marginals. In *Symposium on Computational Geometry (SOCG)*, 2014.

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

[DSS$^+$15] Cynthia Dwork, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. Robust traceability from trace amounts. In *FOCS*. IEEE, 2015.

[DSS16] Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limitations on learning DNFs. In *COLT*, 2016.

[DSSU17] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! a survey of attacks on private data. 2017.

[DTTZ14] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Symposium on Theory of Computing, STOC*, pages 11–20, 2014.

[DY13] Yevgeniy Dodis and Yu Yu. Overcoming weak expectations. In *Theory of Cryptography Conference (TCC)*. 2013.

[GHRU13] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan Ullman. Privately releasing conjunctions and the statistical query barrier. *SIAM J. Comput.*, 42(4):1494–1520, 2013.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Risky traitor tracing and new differential privacy negative results. Cryptology ePrint Archive, Report 2017/1117, 2017.

[GRU12] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *Theory of Cryptography Conference (TCC)*, 2012.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, 2012.

[HLM12] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[HR14] Moritz Hardt and Guy Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Foundations of Computer Science (FOCS)*, 2014.

[HRS12] Moritz Hardt, Guy N. Rothblum, and Rocco A. Servedio. Private data release via learning thresholds. In *Symposium on Discrete Algorithms, (SODA)*, 2012.

[HU14] Moritz Hardt and Jonathan Ullman. Preventing false discovery in interactive data analysis is hard. In *FOCS*. IEEE, 2014.

[Kea93] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. In *Symposium on Theory of Computing (STOC)*. ACM, 1993.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th ACM Symposium on Theory of Computing (STOC)*, 1988.

[KMUZ16] Lucas Kowalczyk, Tal Malkin, Jonathan Ullman, and Mark Zhandry. Strong hardness of privacy from weak traitor tracing. In *Theory of Cryptography Conference (TCC)*, 2016.

[NTZ16] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: The small database and approximate cases. *SIAM J. Comput.*, 45(2):575–616, 2016.

[PV88] Leonard Pitt and Leslie G Valiant. Computational limitations on learning from examples. *Journal of the ACM (JACM)*, 35(4):965–984, 1988.

[RR10] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Symposium on Theory of Computing (STOC)*. ACM, 2010.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *Conference on Computer and Communications Security (CCS)*, 2010.

[SU15] Thomas Steinke and Jonathan Ullman. Interactive fingerprinting codes and the hardness of preventing false discovery. In *Proceedings of The 28th Conference on Learning Theory, COLT*, pages 1588–1628, 2015.

[SU17] Thomas Steinke and Jonathan Ullman. Tight lower bounds for differentially private selection. In *IEEE 58th Annual Symposium on Foundations of Computer Science, FOCS*, pages 634–649, 2017.

[TUV12] Justin Thaler, Jonathan Ullman, and Salil P. Vadhan. Faster algorithms for privately releasing marginals. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2012.

[Ull15] Jonathan Ullman. Private multiplicative weights beyond linear queries. In *PODS*. ACM, 2015.

[Ull16] Jonathan Ullman. Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. *SIAM Journal on Computing*, 45(2):473–496, 2016.

[UV11] Jonathan Ullman and Salil P. Vadhan. PCPs and the hardness of generating private synthetic data. In *Theory of Cryptography Conference (TCC*, 2011.

[Vad16] Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*. Yehuda Lindell, editor, 2016.