Strong Hardness of Privacy from Weak Traitor Tracing

Lucas Kowalczyk* Tal Malkin† Jonathan Ullman‡ Mark Zhandry§

May 30, 2018

Abstract

A central problem in differential privacy is to accurately answer a large family Q of *statistical queries* over a *data universe* X. A statistical query on a dataset $D \in X^n$ asks "what fraction of the elements of D satisfy a given predicate p on X?" Ignoring computational constraints, it is possible to accurately answer exponentially many queries on an exponential size universe while satisfying differential privacy (Blum et al., STOC'08). Dwork et al. (STOC'09) and Boneh and Zhandry (CRYPTO'14) showed that if both Q and X are of polynomial size, then there is an efficient differentially private algorithm that accurately answers all the queries. They also proved that if Q and X are *both* exponentially large, then under a plausible assumption, no efficient algorithm exists.

We show that, under the same assumption, if *either* the number of queries *or* the data universe is of exponential size, then there is no differentially private algorithm that answers all the queries. Specifically, we prove that if one-way functions and indistinguishability obfuscation exist, then:

1. For every n, there is a family Q of $\tilde{O}(n^7)$ queries on a data universe X of size 2^d such that no $\mathrm{poly}(n,d)$ time differentially private algorithm takes a dataset $D \in X^n$ and outputs accurate answers to every query in Q.
2. For every n, there is a family Q of 2^d queries on a data universe X of size $\tilde{O}(n^7)$ such that no $\mathrm{poly}(n,d)$ time differentially private algorithm takes a dataset $D \in X^n$ and outputs accurate answers to every query in Q.

In both cases, the result is nearly quantitatively tight, since there is an efficient differentially private algorithm that answers $\tilde{\Omega}(n^2)$ queries on an exponential size data universe, and one that answers exponentially many queries on a data universe of size $\tilde{\Omega}(n^2)$.

Our proofs build on the connection between hardness results in differential privacy and traitor-tracing schemes (Dwork et al., STOC'09; Ullman, STOC'13). We prove our hardness result for a polynomial size query set (resp., data universe) by showing that they follow from the existence of a special type of traitor-tracing scheme with very short ciphertexts (resp., secret keys), but very weak security guarantees, and then constructing such a scheme.

*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.

§MIT EECS and Princeton University Department of Computer Science. mzhandry@gmail.com.

Contents

1 Introduction

The goal of privacy-preserving data analysis is to release rich statistical information about a sensitive dataset while respecting the privacy of the individuals represented in that dataset. The past decade has seen tremendous progress towards understanding when and how these two competing goals can be reconciled, including surprisingly powerful differentially private algorithms as well as computational and information-theoretic limitations. In this work, we further this agenda by showing a strong new computational bottleneck in differential privacy.

Consider a dataset $D \in X^n$ where each of the n elements is one individual's data, and each individual's data comes from some *data universe* X. We would like to be able to answer sets of *statistical queries* on D, which are queries of the form "What fraction of the individuals in D satisfy some property p?" However, *differential privacy* [DMNS06] requires that we do so in such a way that no individual's data has significant influence on the answers.

If we are content answering a relatively small set of queries Q, then it suffices to perturb the answer to each query with independent noise from an appropriate distribution. This algorithm is simple, very efficient, differentially private, and ensures good accuracy—say, within $\pm.01$ of the true answer—as long as $|Q| \lesssim n^2$ queries [DN03, DN04, BDMN05, DMNS06].

Remarkably, the work of Blum, Ligett, and Roth [BLR13] showed that it is possible to output a summary that allows accurate answers to an *exponential* number of queries—nearly 2^n—while ensuring differential privacy. However, neither their algorithm nor the subsequent improvements [DNR+09, DRV10, RR10, HR10, GRU12, NTZ13, Ull15] are computationally efficient. Specifically, they all require time at least $\mathrm{poly}(n, |X|, |Q|)$ to privately and accurately answer a family of statistical queries Q on a dataset $D \in X^n$. Note that the size of the input is $n \log |X|$ bits, so a computationally efficient algorithm runs in time $\mathrm{poly}(n, \log |X|)$.[1] For example, in the common setting where each individual's data consists of d binary attributes, so $X = \{0,1\}^d$, the size of the input is nd but $|X| = 2^d$. As a result, all known private algorithms for answering arbitrary sets of statistical queries are inefficient if either the number of queries or the size of the data universe is superpolynomial.

This accuracy vs. computation tradeoff has been the subject of extensive study. Dwork et al. [DNR+09] showed that the existence of cryptographic *traitor-tracing schemes* [CFN94] yields a family of statistical queries that cannot be answered accurately and efficiently with differential privacy. Applying recent traitor-tracing schemes [BZ14], we conclude that, under plausible cryptographic assumptions (discussed below), if both the number of queries and the data universe can be superpolynomial, then there is no efficient differentially private algorithm. [Ull13] used variants of traitor-tracing schemes to show that in the interactive setting, where the queries are not fixed but are instead given as input to the algorithm, assuming one-way functions exist, there is no private and efficient algorithm that accurately answers more than $\tilde{O}(n^2)$ statistical queries. All of the algorithms mentioned above work in this interactive setting, but for many applications we only need to answer a fixed family of statistical queries.

Despite the substantial progress, there is still a basic gap in our understanding. The hardness results for Dwork et al. apply if *both* the number of queries and the universe are large. But the known algorithms require exponential time if *either* of these sets is large. Is this necessary? Are there algorithms that run in time $\mathrm{poly}(n, \log |X|, |Q|)$ or $\mathrm{poly}(n, |X|, \log |Q|)$?

[1] It may require exponential time just to describe and evaluate an arbitrary counting query, which would rule out efficiency for reasons that have nothing to do with privacy. In this work, we restrict attention to queries that are efficiently computable in time $\mathrm{poly}(n, \log |X|)$, so they are not the bottleneck in the computation.

Our main result shows that under the same plausible cryptographic assumptions, the answer is no—if either the data universe or the set of queries can be superpolynomially large, then there is some family of statistical queries that cannot be accurately and efficiently answered while ensuring differential privacy.

1.1 Our Results

Our first result shows that if the data universe can be of superpolynomial size then there is some fixed family of polynomially many queries that cannot be efficiently answered under differential privacy. This result shows that the efficient algorithm for answering an arbitrary family of $|Q| \lesssim n^2$ queries by adding independent noise is optimal up to the specific constant in the exponent.

Theorem 1.1 (Hardness for small query sets)**.** *Assume the existence of indistinguishability obfuscation and one-way functions. Let $\lambda \in \mathbb{N}$ be a computation parameter. For any polynomial $n = n(\lambda)$, there is a sequence of pairs $\{(X_\lambda, Q_\lambda)\}$ with $|X_\lambda| = 2^\lambda$ and $|Q_\lambda| = \tilde{O}(n^7)$ such that there is no polynomial time differentially private algorithm that takes a dataset $D \in X_\lambda^n$ and outputs an accurate answer to every query in Q_λ up to an additive error of $\pm 1/3$.*

Our second result shows that, even if the data universe is required to be of polynomial size, there is a fixed set of superpolynomially many queries that cannot be answered efficiently under differential privacy. When we say that an algorithm efficiently answers a set of superpolynomially many queries, we mean that it efficiently outputs a summary such that there is an efficient algorithm for obtaining an accurate answer to any query in the set. For comparison, if $|X| \lesssim n^2$, then there is a simple $\mathrm{poly}(n, |X|)$ time differentially private algorithm that accurately answers superpolynomially many queries.[2] Our result shows that this efficient algorithm is optimal up to the specific constant in the exponent.

Theorem 1.2 (Hardness for small query sets)**.** *Assume the existence of indistinguishability obfuscation and one-way functions. Let $\lambda \in \mathbb{N}$ be a computation parameter. For any polynomial $n = n(\lambda)$, there is a sequence of pairs $\{(X_\lambda, Q_\lambda)\}$ with $|X_\lambda| = \tilde{O}(n^7)$ and $|Q_\lambda| = 2^\lambda$ such that there is no polynomial time differentially private algorithm that takes a dataset $D \in X_\lambda^n$ and outputs an accurate answer to every query in Q_λ up to an additive error of $\pm 1/3$.*

Before we proceed to describe our techniques, we make a few remarks about these results. In both of these results, the constant 1/3 in our result is arbitrary, and can be replaced with any constant smaller than 1/2. We also remark that, when we informally say that an algorithm is differentially private, we mean that it satisfies (ε, δ)-differential privacy for some $\varepsilon = O(1)$ and $\delta = O(1/n)$. These are effectively the largest parameters for which differential privacy is a meaningful notion of privacy. That our hardness results apply to these parameters only makes our results stronger.

[2] The algorithm, sometimes called the *noisy histogram algorithm*, works as follows. First, convert the dataset D to a vector $(D_x)_{x \in X}$ where D_x is the fraction of D's elements that are equal to x. Then, output a vector $\tilde{D} = (\tilde{D}_x)_{x \in X}$ where $\tilde{D}_x$ is equal to D_x plus independent noise from an appropriately scaled Gaussian distribution. To answer a statistical query defined by a predicate p, construct the vector $\vec{p} = (p(x))_{x \in X}$ and compute the answer $\langle \tilde{D}, \vec{p} \rangle$. One can show that this algorithm is differentially private and for any fixed set of statistical queries Q, with high probability, the maximum error is $\tilde{O}(\sqrt{|X| \log |Q|}/n)$. The running time is $\mathrm{poly}(n, |X|)$ to construct $\tilde{D}$ and to evaluate each query.

On Indistinguishability Obfuscation. Indistinguishability obfuscation (iO) has recently become a central cryptographic primitive. The first candidate construction, proposed just a couple years ago [GGH+13], was followed by a flurry of results demonstrating the extreme power and wide applicability of iO (cf., [GGH+13, SW14, BZ14, HSW14, BPW16]). However, the assumption that iO exists is currently poorly understood, and the debate over the plausibility of iO is far from settled. While some specific proposed iO schemes have been attacked [CGH+15, MSZ16], other schemes seem to resist all *currently known* attacks [BMSZ16, GMS16]. We also do not know how to base iO on a solid, simple, natural computational assumption (some attempts based on multilinear maps have been made [GLSW15], but they were broken with respect to all current multilinear map constructions).

Nevertheless, our results are meaningful whether or not iO exists. If iO exists, our results show that certain tasks in differential privacy are intractable. Interestingly, unlike many previous results relying on iO, these conclusions were not previously known to follow from even the much stronger (and in fact, false) assumption of virtual black-box obfuscation. If, on the other hand, iO does not exist, then our results still demonstrate a barrier to progress in differential privacy—such progress would need to *prove* that iO does not exist. Alternatively, our results highlight a possible path toward proving that iO does not exist. We note that other "incompatibility" results are known for iO; for example, iO and certain types of hash functions cannot simultaneously exist [BFM14, BST16].

1.2 Techniques

We prove our results by building on the connection between differentially private algorithms for answering statistical queries and traitor-tracing schemes 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.

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 differentially private.

To instantiate this result, they need a traitor-tracing scheme. Observe that the data universe contains one element for every possible user key, and the set of queries contains one query

for every ciphertext, and we want to minimize the size of these sets. Boneh and Zhandry constructed a traitor-tracing scheme where both the keys and the ciphertexts have length equal to the security parameter λ, which yields hardness for a data universe and query set each of size 2^λ. The main contribution of this work is to show that we can reduce either the number of possible ciphertexts or the number of possible keys to $\text{poly}(n)$ while the other remains of size 2^λ.

Suppose we want to reduce the number of possible ciphertexts to $\text{poly}(n)$. How can we possibly have a secure traitor-tracing scheme with only polynomially many ciphertexts, when even a semantically secure private key encryption scheme requires superpolynomially many ciphertexts? The answer lies in an observation from [Ull13] that in order to show hardness for differential privacy, it suffices to have a traitor-tracing scheme with extremely weak security. First, in the reduction from differential privacy to breaking traitor-tracing the adversary has to produce the pirate decoder using only the coalition's user keys and does not have access to an encryption oracle. Second, the probability that tracing fails only needs to be $o(1/n)$, rather than negligible. Both of these relaxations of the standard definition of traitor-tracing are crucial to making the ciphertext size $\text{poly}(n)$, and as we show, these two relaxations are in fact sufficient. Alternatively, we can use these relaxations also allow us to reduce the key size to $\text{poly}(n)$. We defer the reader to the constructions of Sections 6 and 7 for more details about how we achieve this goal.

1.3 Related Work

Theorem 1.1 should be contrasted with the line of work on answering *width-w marginal queries* under differential privacy [GHRU13, HRS12, TUV12, CTUW14, DNT14]. A width-w marginal query is defined on the data universe $\{0,1\}^\lambda$. It is specified by a set of positions $S \subseteq \{1,\ldots,\lambda\}$ of size w, and a pattern $t \in \{0,1\}^w$ and asks "What fraction of elements of the dataset have each coordinate $j \in S$ set to t_j?" Specifically, Thaler, Ullman, and Vadhan [TUV12], building on the work of Hardt, Rothblum, and Servedio [HRS12] gave an efficient differentially private algorithm for answering $n^{\Omega(\sqrt{w})} \gg n^7$ width-w marginal queries up to an additive error of $\pm.01$. There are also computationally efficient algorithms that answer exponentially many queries from even simpler families like *point queries* and *threshold queries* [BNS13, BNSV15].

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. This result is incomparable to ours, since it applies to a very small and simple family of statistical queries, but necessarily only applies to algorithms that output synthetic data.

Gupta et al. [GHRU13] showed that no algorithm can obtain accurate answers to all marginal queries just by asking a polynomial number of statistical queries on the dataset. Thus, any algorithm that can be implemented using only statistical queries, even one that is not differentially private, can run in polynomial time.

Bun and Zhandry considered the incomparable problem of *differentially private PAC learning* [BZ16] and showed that there is a concept class that is efficiently PAC learnable and inefficiently PAC learnable under differential privacy, but is not efficiently PAC learnable under differential privacy, settling an open question of Kasvisiwanathan et al. [KLN+11], who introduced the model of differentially private PAC learning.

There is also a line of work using *fingerprinting codes* to prove *information-theoretic* lower

bounds on differentially private mechanisms [BUV14, SU15a, DSS+15]. Namely, that if the data universe is of size $\exp(n^2)$, then there is no differentially private algorithm, even a computationally unbounded one, that can answer more than n^2 statistical queries. Fingerprinting codes are essentially the information-theoretic analogue of traitor-tracing schemes, and thus these results are technically related, although the models are incomparable.

Finally, we remark that techniques for proving hardness results in differential privacy have also found applications to the problem of *interactive data analysis* [HU14, SU15b]. The technical core of these results is to show that if an adversary is allowed to ask an online sequence of adaptively chosen statistical queries, then he can not only recover one element of the dataset, but can actually recover every element of the dataset. Doing so rules out any reasonable notion of privacy, and makes many non-private learning tasks impossible. The results are proven using variants of the sorts of traitor-tracing schemes that we study in this work.

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 $D = (D_1, \ldots, D_n)$ 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 subset $T \subseteq S$,

$$\mathbb{P}[A(D) \in T] \leq e^{\varepsilon} \cdot \mathbb{P}[A(D') \in T] + \delta.$$

In this definition, ε, δ may be a function of 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 $D = (D_1, \ldots, D_n) \in X^n$ 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 [Kea98], 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 $0 \le q(D) \le 1$, 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 $Eval(S,q)$.[3]

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

$$\forall q \in Q \quad |q(D) - q(s)| \le \alpha.$$

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 $D \in X^n$,

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

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 $\lambda \in \mathbb{N}$. We then consider a sequence of pairs $\{(X_\lambda, Q_\lambda)\}_{\lambda\in\mathbb{N}}$ where Q_λ is a set of statistical queries on X_λ. We consider databases of size n where $n = n(\lambda)$ is a polynomial. We then consider algorithms A that take as input a dataset X_λ^n and output a summary in S_λ where $\{S_\lambda\}_{\lambda\in\mathbb{N}}$ is a sequence of output ranges. There is an associated evaluator *Eval* that takes a query $q \in Q_\lambda$ and a summary $s \in S_\lambda$ 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 λ. [4] We remark that 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. In this work we restrict attention exclusively to statistical queries that are specified by a circuit of

[3] If we do not restrict the running time of the algorithm, then it is without loss of generality for the algorithm to 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 $|Q|$. Using this framework is crucial, since some of our results concern settings where the number of queries is exponential in the size of the dataset.

[4] The constraint that the evaluator run in polynomial time sounds academic, but is surprisingly crucial. For any Q on X, there is an extremely simple differentially private algorithm that runs in time $\text{poly}(n, |Q|)$ and outputs a summary that is accurate for Q, yet the summary takes time $\text{poly}(|X|, |Q|)$ to evaluate [NTZ13].

size poly($\log|X|$), and thus can be evaluated in time poly($\log|X|$), and so are not the bottleneck in computation. To remind the reader of this fact, we will often say that $\mathcal{Q}$ 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. Also, given a vector $\vec{v} = (v_1, \ldots, v_n) \in X^n$ and an index $i \in [n]$, we will use v_{-i} to denote the vector $\vec{v}_{-i} = (v_1, \ldots, v_{i-1}, \perp, v_{i+1}, \ldots, v_n) \in X^n$ in which the i-th element of $\vec{v}$ is replaced by some unspecified fixed element of X denoted $\perp$. We also say that a function f is *negligible*, and write $f(n) = \mathrm{negl}(n)$, if $f(n) = O(1/n^c)$ for every constant $c > 0$.

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.

3.1 Syntax and Correctness

For a function $n : \mathbb{N} \to \mathbb{N}$ and a sequence $\{K_\lambda, C_\lambda\}_{\lambda \in \mathbb{N}}$, an $(n, \{K_\lambda, C_\lambda\})$-*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 λ, runs in time poly(λ), and outputs $n = n(\lambda)$ secret *user keys* $sk_1, \ldots, sk_n \in K_\lambda$ and a secret *master key* mk. We will write $\vec{k} = (sk_1, \ldots, sk_n, mk)$ to denote the set of keys.
- Enc takes as input a master key mk and an *index* $i \in \{0, 1, \ldots, n\}$, and outputs a ciphertext $c \in C_\lambda$. If $c \leftarrow_R \mathsf{Enc}(j, mk)$ 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 $\vec{k}$ are generated by Setup, then for any pair i, j, $\mathsf{Dec}(sk_i, \mathsf{Enc}(mk, j)) = \mathbb{I}\{i \leq j\}$. 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 $(n, \{K_\lambda, C_\lambda\})$-traitor-tracing scheme is *perfectly correct* if for every $\lambda \in \mathbb{N}$, and every $i, j \in \{0, 1, \ldots, n\}$

$$\mathop{\mathbb{P}}_{\vec{k}=\mathsf{Setup}(\lambda), c=\mathsf{Enc}(mk,j)} [\mathsf{Dec}(sk_i, c) = \mathbb{I}\{i \leq j\}] = 1.$$

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 $\lambda \in \mathbb{N}$, every $i^* \in [n(\lambda)]$, and every adversary A with running time $\mathrm{poly}(\lambda)$,

$$\mathop{\mathbb{P}}_{\vec{k}=\mathsf{Setup}(\lambda),\, S=A(sk_{-i^*})}\left[\mathbb{P}\left[S(\mathsf{Enc}(mk,i^*))=1\right]-\mathbb{P}\left[S(\mathsf{Enc}(mk,i^*-1))=1\right]>\frac{1}{2en}\right]\leq\frac{1}{4en} \tag{1}$$

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.

3.2.1 The Index-Hiding and Two-Index-Hiding Games

While Definition 3.2 is the most natural, in this section we consider some related ways of defining security that will be easier to work with when we construct and analyze our schemes. Consider the following **IndexHiding** game.

The challenger generates keys $\vec{k} = (sk_1, \ldots, sk_n, mk) \leftarrow_{\mathrm{R}} \mathsf{Setup}(\lambda)$.
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 $i^* - b$, $c \leftarrow_{\mathrm{R}} \mathsf{Enc}(mk, i^* - b)$
The adversary makes a guess $b' = S(c)$

Figure 1: **IndexHiding**$[i^*]$

Let **IndexHiding**$[i^*, \vec{k}, S]$ be the game **IndexHiding**$[i^*]$ where we fix the choices of $\vec{k}$ and S. Also, define

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

so that

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

Then the following is equivalent to (1) in Definition 3.2 as

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

In order to prove that our schemes satisfy weak index-hiding security, we will go through an intermediate notion that we call two-index-hiding security. To see why this is useful, In our constructions it will be fairly easy to prove that $\mathrm{Adv}[i^*]$ is small, but because $\mathrm{Adv}[i^*, \vec{k}, S]$ can be positive or negative, that alone is not enough to establish (2). Thus, in order to establish (2) we will analyze the following variant of the index-hiding game.

The challenger generates keys $\vec{k} = (sk_1, \ldots, sk_n, mk) \leftarrow_R \mathsf{Setup}$.
The adversary A is given keys sk_{-i^*} and outputs a decryption program S.
Choose $b_0 \leftarrow_R \{0,1\}$ and $b_1 \leftarrow_R \{0,1\}$ independently.
Let $c_0 \leftarrow_R \mathsf{Enc}(i^* - b_0; mk)$ and $c_1 \leftarrow_R \mathsf{Enc}(i^* - b_1; mk)$.
Let $b' = S(c_0, c_1)$.

Figure 2: **TwoIndexHiding**$[i^*]$

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

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

$$\mathrm{TwoAdv}[i^*, \vec{k}, S] = \mathop{\mathbb{P}}_{\mathbf{TwoIndexHiding}[i^*, \vec{k}, S]}[b' = b_0 \oplus b_1] - \frac{1}{2}$$

so that

$$\mathop{\mathbb{P}}_{\mathbf{TwoIndexHiding}[i^*]}[b' = b_0 \oplus b_1] - \frac{1}{2} = \mathop{\mathbb{E}}_{\vec{k} = \mathsf{Setup}(\lambda), S = A(sk_{-i^*})}\left[\mathrm{TwoAdv}[i^*, \vec{k}, S]\right]$$

The crucial feature is that if we can bound the expectation of TwoAdv then we get a bound on the expectation of Adv^2. Since Adv^2 is always positive, we can apply Markov's inequality to establish (2). Formally, we have the following claim.

Claim 3.3. *Suppose that for every efficient adversary A, $\lambda \in \mathbb{N}$, and index $i^* \in [n(\lambda)]$,*

$$\mathrm{TwoAdv}[i^*] \leq \varepsilon.$$

Then for every efficient adversary A, $\lambda \in \mathbb{N}$, and index $i^* \in [n(\lambda)]$,

$$\mathop{\mathbb{E}}_{\substack{\vec{k} = \mathsf{Setup}(\lambda), \\ S \leftarrow A(sk_{-i^*})}}\left[\mathrm{Adv}[i^*, \vec{k}, S]^2\right] \leq \frac{\varepsilon}{2}. \tag{3}$$

Proof. Given any adversary A in the **IndexHiding** game, consider the following adversary A_2 in the **TwoIndexHiding** game, which, when given a set of keys, runs A with the same keys to get program S_A, then creates and outputs the program S_{A_2}, which on input c_0, c_1, runs S on c_0 to get output b'_0, runs S on c_1 to get output b'_1, then outputs $b' = b'_0 \oplus b'_1$. Then, for this A_2,

$$
\begin{aligned}
\mathrm{TwoAdv}[i^*] &= \mathop{\mathbb{E}}_{\substack{\vec{k}=\mathsf{Setup}(\lambda),\\ S_{A_2}\leftarrow A_2(sk_{-i^*})}} \left[\mathrm{TwoAdv}[i^*,\vec{k},S_{A_2}]\right] \\
&= \mathop{\mathbb{E}}_{\substack{\vec{k}=\mathsf{Setup}(\lambda),\\ S_{A_2}\leftarrow A_2(sk_{-i^*})}} \left[\Pr_{\substack{b_i\leftarrow_R\{0,1\},\\ c_i\leftarrow\mathsf{Enc}(i^*-b_i)}} [b' = b_0\oplus b_1 \ :\ b' = S_{A_2}(c_0,c_1)] - \frac{1}{2}\right] \\
&= \mathop{\mathbb{E}}_{\substack{\vec{k}=\mathsf{Setup}(\lambda),\\ S_{A}\leftarrow A(sk_{-i^*})}} \left[\Pr_{\substack{b_i\leftarrow_R\{0,1\},\\ c_i\leftarrow\mathsf{Enc}(i^*-b_i)}} [b'_0\oplus b'_1 = b_0\oplus b_1 \ :\ b'_i = S_{A}(c_i)] - \frac{1}{2}\right] \\
&= \mathop{\mathbb{E}}_{\substack{\vec{k}=\mathsf{Setup}(\lambda),\\ S_{A}\leftarrow A(sk_{-i^*})}} \left[\Pr_{\substack{b_i\leftarrow_R\{0,1\},\\ c_i\leftarrow\mathsf{Enc}(i^*-b_i),\\ b'_i=S_A(c_i)}} [(b'_0 = b_0)\wedge(b'_1 = b_1)] + \Pr_{\substack{b_i\leftarrow_R\{0,1\},\\ c_i\leftarrow\mathsf{Enc}(i^*-b_i),\\ b'_i=S_A(c_i)}} [(b'_0 \neq b_0)\wedge(b'_1 \neq b_1)] - \frac{1}{2}\right] \\
&= \mathop{\mathbb{E}}_{\substack{\vec{k}=\mathsf{Setup}(\lambda),\\ S_{A}\leftarrow A(sk_{-i^*})}} \left[(\frac{1}{2} + \mathrm{Adv}[i^*,\vec{k},S_A])^2 + (\frac{1}{2} - \mathrm{Adv}[i^*,\vec{k},S_A])^2 - \frac{1}{2}\right] \\
&= 2\cdot \mathop{\mathbb{E}}_{\substack{\vec{k}=\mathsf{Setup}(\lambda),\\ S_{A}\leftarrow A(sk_{-i^*})}} \left[\mathrm{Adv}[i^*,\vec{k},S_A]^2\right]
\end{aligned}
$$

So if every efficient adversary A', $\lambda\in\mathbb{N}$, and index $i^*\in[n(\lambda)]$ satisfies:

$$\mathrm{TwoAdv}[i^*]\leq\varepsilon.$$

then this holds for this A_2's $\mathrm{TwoAdv}[i^*] = 2\cdot \mathop{\mathbb{E}}_{\substack{\vec{k}=\mathsf{Setup}(\lambda),\\ S_{A}\leftarrow A(sk_{-i^*})}} \left[\mathrm{Adv}[i^*,\vec{k},S_A]^2\right]$, which means that

$$\mathop{\mathbb{E}}_{\substack{\vec{k}=\mathsf{Setup}(\lambda),\\ S_{A}\leftarrow A(sk_{-i^*})}} \left[\mathrm{Adv}[i^*,\vec{k},S_A]^2\right]\leq\frac{\varepsilon}{2}$$

□

Using this claim we can prove the following lemma.

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

$$\mathrm{TwoAdv}[i^*]\leq\frac{1}{300n^3}.$$

Then Π *satisfies weak index-hiding security.*

Proof. By applying Claim 3.3 to the assumption of the lemma, we have that for every efficient adversary A,

$$\mathop{\mathbb{E}}_{\vec{k}=\mathsf{Setup}(\lambda),S=A(sk_{-i^*})} \left[\mathrm{Adv}[i^*,\vec{k},S]^2\right]\leq\frac{1}{600n^3}$$

Now we have

$$\mathop{\mathbb{E}}_{\vec{k}=\mathsf{Setup}(\lambda),S=A(sk_{-i^*})}\left[\mathrm{Adv}[i^*,\vec{k},S]^2\right] \le \frac{1}{600n^3}$$

$$\implies \mathop{\mathbb{P}}_{\vec{k}=\mathsf{Setup}(\lambda),S=A(sk_{-i^*})}\left[\mathrm{Adv}[i^*,\vec{k},S]^2 > \frac{1}{(4en)^2}\right] \le \frac{(4en)^2}{600n^3} \le \frac{1}{4en} \qquad \text{(Markov's Inequality)}$$

$$\implies \mathop{\mathbb{P}}_{\vec{k}=\mathsf{Setup}(\lambda),S=A(sk_{-i^*})}\left[\mathrm{Adv}[i^*,\vec{k},S] > \frac{1}{4en}\right] \le \frac{1}{4en}$$

To complete the proof, observe that this final condition is equivalent to the definition of weak index-hiding security (Definition 3.2). □

In light of this lemma, we will focus on proving that the schemes we construct in the following sections satisfying the condition

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

which will be easier than directly establishing Definition 3.2.

4 Hardness of Differential Privacy from Traitor Tracing

In this section we prove that traitor-tracing scheme satisfying perfect correctness and index-hiding security yields a family of statistical queries that cannot be answered accurately by an efficient differentially private algorithm. The proof is a fairly straightforward adaptation of the proofs in Dwork et al. [DNR+09] and Ullman [Ull13] that various sorts of traitor-tracing schemes imply hardness results for differential privacy. We include the result for completeness, and to verify that our very weak definition of traitor-tracing is sufficient to prove hardness of differential privacy.

Theorem 4.1. *Suppose there is an $(n,\{K_\lambda,C_\lambda\})$-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 $\{X_\lambda,Q_\lambda\}_{\lambda\in\mathbb{N}}$ where Q_λ is a set of statistical queries on X_λ, $|Q_\lambda|=|C_\lambda|$, and $|X_\lambda|=|K_\lambda|$ such that there is no algorithm A that is simultaneously,*

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

Theorem 1.1 and 1.2 in the introduction follow by combining Theorem 4.1 above with the constructions of traitor-tracing schemes in Sections 6 and 7. The proof of Theorem 4.1 closely follows the proofs in Dwork et al. [DNR+09] and Ullman [Ull13]. We give the proof both for completeness and to verify that our definition of traitor-tracing suffices to establish the hardness of differential privacy.

Proof. Let $\Pi=(\mathsf{Setup},\mathsf{Enc},\mathsf{Dec})$ be the promised $(n,\{K_\lambda,C_\lambda\})$ traitor-tracing scheme. For every $\lambda\in\mathbb{N}$, we can define a distribution on datasets $D\in X_\lambda^{n(\lambda)}$ as follows. Run $\mathsf{Setup}(\lambda)$ to obtain

$n = n(\lambda)$ secret user keys $sk_1, \ldots, sk_n \in K_\lambda$ and a master secret key mk. Let the dataset be $D = (sk_1, \ldots, sk_n) \in X_\lambda^n$ where we define the data universe $X_\lambda = K_\lambda$. Abusing notation, we'll write $(D, mk) \leftarrow_R \mathsf{Setup}(\lambda)$.

Now we define the family of queries Q_λ on X_λ as follows. For every ciphertext $c \in C_\lambda$, we define the predicate $q_c \in Q_\lambda$ to take as input a user key $sk_i \in K_\lambda$ and output $\mathsf{Dec}(sk_i, c)$. That is,

$$Q_\lambda = \{q_c(sk) = \mathsf{Dec}(sk, c) \mid c \in C_\lambda\}.$$

Recall that, by the definition of a statistical query, for a dataset $D = (sk_1, \ldots, sk_n)$, we have

$$q_c(D) = (1/n) \sum_{i=1}^{n} \mathsf{Dec}(sk_i, c).$$

Suppose there is an algorithm A that is computationally efficient and is $(1/3, 1/2n)$-accurate for Q_λ given a dataset $D \in X_\lambda^n$. We will show that A cannot satisfy $(1, 1/4n)$-differential privacy. By accuracy, for every $\lambda \in \mathbb{N}$ and every fixed dataset $D \in X_\lambda^n$, with probability at least $1 - 1/2n$, $A(D)$ outputs a summary $S \in \mathcal{S}_\lambda$ that is 1/3-accurate for Q_λ on D. That is, for every $D \in X_\lambda^n$, with probability at least $1 - 1/2n$,

$$\forall q_c \in Q_\lambda \quad |q_c(D) - q_c(S)| \leq 1/3. \tag{4}$$

Suppose that S is indeed 1/3-accurate. By perfect correctness of the traitor-tracing scheme (Definition 3.1), and the definition of Q, we have that since $(D, mk) = \mathsf{Setup}(\lambda)$,

$$(c = \mathsf{Enc}(mk, 0)) \Longrightarrow (q_c(D) = 0) \qquad (c = \mathsf{Enc}(mk, n)) \Longrightarrow (q_c(D) = 1). \tag{5}$$

Combining Equations (4) and (5), we have that if $(D, mk) = \mathsf{Setup}(\lambda)$, $S \leftarrow_R A(D)$, and S is 1/3-accurate, then we have both

$$\mathop{\mathbb{P}}_{c \leftarrow_R \mathsf{Enc}(mk,0)} [q_c(S) \leq 1/3] = 1 \qquad \mathop{\mathbb{P}}_{c \leftarrow_R \mathsf{Enc}(mk,n)} [q_c(S) \leq 1/3] = 0$$

Thus, for every (D, mk) and S that is 1/3-accurate, there exists an index $i \in \{1, \ldots, n\}$ such that

$$\left| \mathop{\mathbb{P}}_{c \leftarrow_R \mathsf{Enc}(mk,i)} [q_c(S) \leq 1/3] - \mathop{\mathbb{P}}_{c \leftarrow_R \mathsf{Enc}(mk,i-1)} [q_c(S) \leq 1/3] \right| > \frac{1}{n} \tag{6}$$

By averaging, using the fact that S is 1/3-accurate with probability at least $1 - 1/2n$, there must exist an index $i^* \in \{1, \ldots, n\}$ such that

$$\mathop{\mathbb{P}}_{\substack{(D,mk) = \mathsf{Setup}(\lambda), \\ S \leftarrow_R A(D)}} \left[\left| \mathop{\mathbb{P}}_{c \leftarrow_R \mathsf{Enc}(mk,i^*)} [q_c(S) \leq 1/3] - \mathop{\mathbb{P}}_{c \leftarrow_R \mathsf{Enc}(mk,i^*-1)} [q_c(S) \leq 1/3] \right| > \frac{1}{n} \right] \geq \frac{1}{n} \cdot (1 - \frac{1}{2n}) \geq \frac{1}{2n} \tag{7}$$

Assume, for the sake of contradiction that A is $(1, 1/4n)$-differentially private. For a given i, mk, D, let $\mathcal{S}_{i,mk,D} \subseteq \mathcal{S}_\lambda$ be the set of summaries such that (6) holds. Then, by (7), we have

$$\mathop{\mathbb{P}}_{(D,mk) \leftarrow_R \mathsf{Setup}(\lambda)} [A(D) \in \mathcal{S}_{i^*,mk,D}] \geq \frac{1}{2n}.$$

By differential privacy of A, we have

$$\mathop{\mathbb{P}}_{(D,mk) \leftarrow_R \mathsf{Setup}} [A(D_{-i^*}) \in \mathcal{S}_{i^*,mk,D}] \geq \frac{1}{e} \left(\frac{1}{2n} - \frac{1}{4n} \right) = \frac{1}{4en}$$

Thus, by our definition of $\mathcal{S}_{i^*,mk}$, and by averaging over $(D, mk) \leftarrow_{\mathrm{R}} \mathsf{Setup}(\lambda)$, we have

$$\mathop{\mathbb{P}}_{(D,mk)=\mathsf{Setup},S\leftarrow_{\mathrm{R}}A(D_{-i^*})}\left[\left|\mathop{\mathbb{P}}_{c\leftarrow_{\mathrm{R}}\mathsf{Enc}(mk,i^*)}[q_c(S)\le 1/3]-\mathop{\mathbb{P}}_{c\leftarrow_{\mathrm{R}}\mathsf{Enc}(mk,i^*-1)}[q_c(S)\le 1/3]\right|>\frac{1}{n}\right]\ge\frac{1}{4en} \tag{8}$$

But this violates the index hiding property of the traitor tracing scheme. Specifically, if we consider an adversary for the traitor tracing scheme that runs A on the keys sk_{-i^*} to obtain a summary S, then decrypts a ciphertext c by computing $q_c(S)$ and rounding the answer to $\{0,1\}$, then by (8) this adversary violates index-hiding security (Definition 3.2).

Thus we have obtained a contradiction showing that A is not $(1, 1/4n)$-differentially private. This completes the proof. □

5 Cryptographic Primitives

5.1 Standard Tools

We will make use of a few standard cryptographic and information-theoretic primitives. We will define these primitives for completeness and to set notation and terminology.

Almost Pairwise Independent Hash Families. A hash family is a family of functions $\mathcal{H}_s = \{h : [s] \to \{0,1\}\}$. To avoid notational clutter, we will use the notation $h \leftarrow_{\mathrm{R}} \mathcal{H}$ to denote the operation of choosing a random function from $\mathcal{H}$ and will not explicitly write the seed for the function. We will use $|h|$ to denote the seed length for the function and require that h can be evaluated in time poly$(|h|)$.

Definition 5.1. A family of functions $\mathcal{H}_s = \{h : [T] \to [K]\}$ is *δ-almost pairwise independent* if for every two distinct points $x_0, x_1 \in [T]$, and every $y_0, y_1 \in [K]$,

$$\mathop{\mathbb{P}}_{h\leftarrow_{\mathrm{R}}\mathcal{H}}[h(x_0)=y_0 \wedge h(x_1)=y_1]=\frac{1}{K^2}+\delta.$$

For every s, there exists a pairwise independent hash family $\mathcal{H}_s = \{h : [s] \to \{0,1\}\}$ such that $|h| = O(\log(s))$ for every $h \in \mathcal{H}$.

Pseudorandom Generators. A pseudorandom generator $\mathsf{PRG} : \{0,1\}^{\lambda/2} \to \{0,1\}^{\lambda}$ is a function such that $\mathsf{PRG}(U_{\lambda/2}) \approx_{\mathrm{negl}(\lambda)} U_\lambda$. In this definition, U_b denotes the uniform distribution on $\{0,1\}^b$. Pseudorandom generators exist under the minimal assumption that one-way functions exist.

Pseudorandom Function Families. A pseudorandom function family is a family of functions $\mathcal{F}_\lambda = \{\mathsf{PRF} : [m(\lambda)] \to [n(\lambda)]\}$. To avoid notational clutter, we will use the notation $\mathsf{PRF} \leftarrow_{\mathrm{R}} \mathcal{F}_\lambda$ to denote the operation of choosing a random function from $\mathcal{F}_\lambda$ and not explicitly write the seed for the function. We will use $|\mathsf{PRF}|$ to denote the description length for the function. We require that $|\mathsf{PRF}| = \mathrm{poly}(\lambda)$ and that PRF can be evaluated in time poly$(|\mathsf{PRF}|)$.

Security requires that oracle access to $\mathsf{PRF} \leftarrow_{\mathrm{R}} \mathcal{F}_\lambda$ is indistinguishable from oracle access to a random function. Specifically, for all probabilistic polynomial-time algorithms D,

$$\left|\Pr_{\mathsf{PRF}\leftarrow_{\mathrm{R}}\mathcal{F}_\lambda}[D^{\mathsf{PRF}()}(1^\lambda)=1]-\Pr_{f\leftarrow_{\mathrm{R}}\{f:[m]\to[n]\}}[D^{f()}(1^\lambda)=1]\right|<\varepsilon(\lambda)$$

for some negligible function ε.

Under the minimal assumption that one-way functions exist, for every pair of functions m, n that are at most exponential, for every $\lambda \in \mathbb{N}$, there is a family of pseudorandom functions $\mathcal{F}_\lambda = \{\mathsf{PRF} : [m(\lambda)] \to [n(\lambda)]\}$ such that $|\mathsf{PRF}| = \mathrm{poly}(\lambda)$.

A pseudorandom function family is δ-almost pairwise independent for $\delta = \mathrm{negl}(\lambda)$.

5.2 Puncturable Pseudorandom Functions

A pseudorandom function family $\mathcal{F}_\lambda = \{\mathsf{PRF} : [m] \to [n]\}$ is *puncturable* if there is a deterministic procedure Puncture that takes as input $\mathsf{PRF} \in \mathcal{F}_\lambda$ and $x^* \in [m]$ and outputs a new function $\mathsf{PRF}^{\{x^*\}} : [m] \to [n]$ such that

$$\mathsf{PRF}^{\{x^*\}}(x) = \begin{cases} \mathsf{PRF}(x) & \text{if } x \neq x^* \\ \bot & \text{if } x = x^* \end{cases}$$

The definition of security for a punctured pseudorandom function states that for any x^*, given the punctured function $\mathsf{PRF}^{\{x^*\}}$, the missing value $\mathsf{PRF}(x^*)$ is computationally unpredictable. Specifically, we define the following game **Puncture** to capture the desired security property.

The challenger chooses $\mathsf{PRF} \leftarrow_{\mathrm{R}} \mathcal{F}_\lambda$
The challenger chooses uniform random bit $b \in \{0, 1\}$, and samples

$$y_0 \leftarrow_{\mathrm{R}} \mathsf{PRF}(x^*), \quad y_1 \leftarrow_{\mathrm{R}} [n].$$

The challenger punctures PRF at x^*, obtaining $\mathsf{PRF}^{\{x^*\}}$.
The adversary is given $(y_b, \mathsf{PRF}^{\{x^*\}})$ and outputs a bit b'.

Figure 3: **Puncture**$[x^*]$

Definition 5.2 (Puncturing Secure PRF)**.** A pseudorandom function family $\mathcal{F}_\lambda = \{\mathsf{PRF} : [m] \to [n]\}$ is ε-*puncturing secure* if for every $x^* \in [m]$,

$$\mathop{\mathbb{P}}_{\mathbf{Puncture}[x^*]}[b' = b] \leq \frac{1}{2} + \varepsilon.$$

5.3 Twice Puncturable PRFs

A *twice puncturable PRF* is a pair of algorithms (*PRFSetup*, Puncture).

- *PRFSetup* is a randomized algorithm that takes a security parameter λ and outputs a function $\mathsf{PRF} : [m] \to [n]$ where $m = m(\lambda)$ and $n = n(\lambda)$ are parameters of the construction. Technically, the function is parameterized by a seed of length λ, however for notational simplicity we will ignore the seed and simply use PRF to denote this function. Formally $\mathsf{PRF} \leftarrow_{\mathrm{R}} \mathit{PRFSetup}(\lambda)$.

- Puncture is a deterministic algorithm that takes a PRF and a pair of inputs $x_0, x_1 \in [m]$ and outputs a new function $\mathsf{PRF}^{\{x_0,x_1\}} : [m] \to [n]$ such that

$$\mathsf{PRF}^{\{x_0,x_1\}} = \begin{cases} \mathsf{PRF}(x) & \text{if } x \notin \{x_0, x_1\} \\ \perp & \text{if } x \in \{x_0, x_1\} \end{cases}$$

  Formally, $\mathsf{PRF}^{\{x_0,x_1\}} = \mathsf{Puncture}(\mathsf{PRF}, x_0, x_1)$.

In what follows we will always assume that m and n are polynomial in the security parameter and that $m = \omega(n \log(n))$.

In addition to requiring that this family of functions satisfies the standard notion of cryptographic pseudorandomness, we will now define a new security property for twice puncturable PRFs, called *input matching indistinguishability*. For any two distinct outputs $y_0, y_1 \in [n], y_0 \neq y_1$, consider the following game.

The challenger chooses PRF such that $\forall y \in [n]$, $\mathsf{PRF}^{-1}(y) \neq \emptyset$.
The challenger chooses independent random bits $b_0, b_1 \in \{0,1\}$, and samples

$$x_0 \leftarrow_{\mathrm{R}} \mathsf{PRF}^{-1}(y_{b_0}), \quad x_1 \leftarrow_{\mathrm{R}} \mathsf{PRF}^{-1}(y_{b_1}).$$

The challenger punctures PRF at x_0, x_1, obtaining $\mathsf{PRF}^{\{x_0,x_1\}}$.
The adversary is given $(x_0, x_1, \mathsf{PRF}^{\{x_0,x_1\}})$ and outputs a bit b'.

Figure 4: $\mathsf{InputMatching}[y_0, y_1]$

Notice that in this game, we have assured that every $y \in [n]$ has a preimage under PRF. We need this condition to make the next step of sampling random preimages well defined. Technically, it would suffice to have a preimage only for y_{b_0} and y_{b_1}, but for simplicity we will assume that every possible output has a preimage. When $f : [m] \to [n]$ is a random function, the probability that some output has no preimage is at most $n \cdot \exp(-\Omega(m/n))$ which is negligible when $m = \omega(n \log(n))$. Since m, n are assumed to be a polynomial in the security parameter, we can efficiently check if every output has a preimage, thus if PRF is pseudorandom it must also be the case that every output has a preimage with high probability. Since we can efficiently check whether or not every output has a preimage under PRF, and this event occurs with all but negligible probability, we can efficiently sample the pseudorandom function in the first step of $\mathsf{InputMatching}[y_0, y_1]$.

Definition 5.3 (Input-Matching Secure PRF)**.** A function family $\{\mathsf{PRF} : [m] \to [n]\}$ is *ε-input-matching secure* if the function family is a secure pseudorandom function and additionally for every $y_0, y_1 \in [n]$ with $y_0 \neq y_1$,

$$\mathop{\mathbb{P}}_{\mathsf{InputMatching}[y_0,y_1]}[b' = b_0 \oplus b_1] \leq \frac{1}{2} + \varepsilon.$$

In Appendix A we will show that input-matching secure twice puncturable pseudorandom functions exist with suitable parameters.

Theorem 5.4. *Assuming the existence of one-way functions, if m, n are polynomials such that $m = \omega(n \log(n))$, then there exists a pseudorandom function family $\mathcal{F}_\lambda = \{\mathsf{PRF} : [m(\lambda)] \to [n(\lambda)]\}$ that is twice puncturable and is $\tilde{O}(\sqrt{n/m})$-input-matching secure.*

5.4 Indistinguishability Obfuscation

We use the following formulation of Garg et al. [GGH+13] for indistinguishability obfuscation:

Definition 5.5 (Indistinguishability Obfuscation)**.** A *indistinguishability obfuscator* O for a circuit class $\{\mathcal{C}_\lambda\}$ is a probabilistic polynomial-time uniform algorithm satisfying the following conditions:

1. $\mathsf{O}(\lambda, C)$ preserves the functionality of C. That is, for any $C \in \mathcal{C}_\lambda$, if we compute $C' = \mathsf{O}(\lambda, C)$, then $C'(x) = C(x)$ for all inputs x.
2. For any λ and any two circuits C_0, C_1 with the same functionality, the circuits $\mathsf{O}(\lambda, C_0)$ and $\mathsf{O}(\lambda, C_1)$ are indistinguishable. More precisely, for all pairs of probabilistic polynomial-time adversaries (Samp, D), if

$$\Pr_{(C_0, C_1, \sigma) \leftarrow \mathrm{Samp}(\lambda)}[(\forall x),\ C_0(x) = C_1(x)] > 1 - \mathrm{negl}(\lambda)$$

   then

$$|\Pr[D(\sigma, \mathsf{O}(\lambda, C_0)) = 1] - \Pr[D(\sigma, \mathsf{O}(\lambda, C_1)) = 1]| < \mathrm{negl}(\lambda)$$

The circuit classes we are interested in are polynomial-size circuits - that is, when $\mathcal{C}_\lambda$ is the collection of all circuits of size at most λ.

When clear from context, we will often drop λ as an input to O and as a subscript for $\mathcal{C}$.

6 A Weak Traitor-Tracing Scheme with Very Short Ciphertexts

In this section we construct a traitor-tracing scheme for n users where the key length is polynomial in the security parameter λ and the ciphertext length is only $O(\log(n))$. This scheme will be used to establish our hardness result for differential privacy when the data universe can be exponentially large but the family of queries has only polynomial size.

6.1 Construction

Let $n = \mathrm{poly}(\lambda)$ denote the number of users for the scheme. Let $m = \tilde{O}(n^7)$ be a parameter. Our construction will rely on the following primitives:

- A pseudorandom generator $\mathsf{PRG} : \{0,1\}^{\lambda/2} \to \{0,1\}^\lambda$.
- A puncturable pseudorandom function family $\mathcal{F}_{\lambda,sk} = \left\{\mathsf{PRF}_{sk} : [n] \to \{0,1\}^\lambda\right\}$.
- A twice-puncturable pseudorandom function family $\mathcal{F}_{\lambda,\mathsf{Enc}} = \{\mathsf{PRF}_{\mathsf{Enc}} : [m] \to [n]\}$.
- An iO scheme $\mathsf{Obfuscate}$.

Theorem 6.1. *Assuming the existence of one-way functions and indistinguishability obfuscation. For every polynomial n, the scheme $\Pi_{\mathrm{short-ctext}}$ is an (n, d, ℓ)-traitor-tracing scheme for $d = \mathrm{poly}(\lambda)$ and $2^\ell = \tilde{O}(n^7)$ and satisfies:*

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

Combining this theorem with Lemma 3.4 and Theorem 4.1 establishes Theorem 1.1 in the introduction.

$\mathsf{Setup}(\lambda)$:
Choose $\mathsf{PRF}_{sk} \leftarrow_R \mathcal{F}_{\lambda,sk}$
Choose $\mathsf{PRF}_{\mathsf{Enc}} \leftarrow_R \mathcal{F}_{\lambda,\mathsf{Enc}}$ such that for every $i \in [n]$, $\mathsf{PRF}_{\mathsf{Enc}}^{-1}(i) \neq \emptyset$
For $i = 1, \ldots, n$, let $s_i = \mathsf{PRF}_{sk}(i)$.
Let $\mathsf{O} \leftarrow_R \mathsf{Obfuscate}(\mathsf{P}_{\mathsf{PRF}_{sk},\mathsf{PRF}_{\mathsf{Enc}}})$.
Let each user's secret key be $sk_i = (i, s_i, \mathsf{O})$
Let the master key be $mk = \mathsf{PRF}_{\mathsf{Enc}}$.

$\mathsf{Enc}(j, mk = \mathsf{PRF}_{\mathsf{Enc}})$:
Let c be chosen uniformly from $\mathsf{PRF}_{\mathsf{Enc}}^{-1}(j)$.
Output c.

$\mathsf{Dec}(sk_i = (i, s_i, \mathsf{O}), c)$:
Output $\mathsf{O}(c, i, s_i)$.

$\mathsf{P}_{\mathsf{PRF}_{sk},\mathsf{PRF}_{\mathsf{Enc}}}(c, i, s)$:
If $\mathsf{PRG}(s) \neq \mathsf{PRG}(\mathsf{PRF}_{sk}(i))$, halt and output $\perp$.
Output $\mathbb{I}\{i \leq \mathsf{PRF}_{\mathsf{Enc}}(c)\}$.

Figure 5: Our scheme $\Pi_{\mathrm{short-ctext}}$.

Parameters

First we verify that $\Pi_{\mathrm{short-ctext}}$ is an (n, d, ℓ)-traitor-tracing scheme for the desired parameters. Observe that the length of the secret keys is $\log(n)+\lambda+|\mathsf{O}|$. By the efficiency of the pseudorandom functions and the specification of P, the running time of P is $\mathrm{poly}(\lambda + \log(n))$. Thus, by the efficiency of $\mathsf{Obfuscate}$, $|\mathsf{O}| = \mathrm{poly}(\lambda + \log(n))$. Therefore the total key length is $\mathrm{poly}(\lambda + \log(n))$. Since n is assumed to be a polynomial in λ, we have that the secret keys have length $d = \mathrm{poly}(\lambda)$ as desired. By construction, the ciphertext is an element of $[m]$. Thus, since $m = \tilde{O}(n^7)$ the ciphertexts length ℓ satisfies $2^\ell = \tilde{O}(n^7)$ as desired.

6.2 Proof of Weak Index-Hiding Security

In light of Lemma 3.4, in order to prove that the scheme satisfies weak index-hiding security, it suffices to show that for every sufficiently large $\lambda \in \mathbb{N}$, and every $i^* \in [n(\lambda)]$,

$$\mathop{\mathbb{P}}_{\mathbf{TwoIndexHiding}[i^*]}\left[b' = b_0 \oplus b_1\right] - \frac{1}{2} = o(1/n^3).$$

We will demonstrate this using a series of hybrids to reduce security of the scheme in the **TwoIndexHiding** game to input-matching security of the pseudorandom function family $\mathsf{PRF}_{\lambda,\mathsf{Enc}}$.

Before we proceed with the argument, we remark a bit on how we will present the hybrids. Note that the view of the adversary consists of the keys sk_{-i^*}. Each of these keys is of the form (i, s_i, O) where O is an obfuscation of the same program P. Thus, for brevity, we will discuss only how we modify the construction of the program P and it will be understood that each user's key will consist of an obfuscation of this modified program. We will also rely crucially on the fact

that, because the challenge ciphertexts depend only on the master key mk, we can generate the challenge ciphertexts c_0 and c_1 can be generated before the users' secret keys $sk_1, \ldots, sk_n$. Thus, we will be justified when we modify P in a manner that depends on the challenge ciphertexts and include an obfuscation of this program in the users' secret keys. We also remark that we highlight the changes in the hybrids in green.

Breaking the decryption program for challenge index

We use a series of hybrids to ensure that the obfuscated program reveals no information about the secret s_{i^*} for the specified user i^*. First, we modify the program by hardcoding the secret s_{i^*} into the program. The obfuscated versions of P and P^1 are indistinguishable because the

$\mathsf{P}^1_{\mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}, i^*, x^*}(c, i, s)$:
If $i = i^*$ and $\mathsf{PRG}(s) \neq x^*$, halt and output $\perp$.
If $i \neq i^*$ and $\mathsf{PRG}(s) \neq \mathsf{PRG}(\mathsf{PRF}_{sk}^{\{i^*\}}(i))$, halt and output $\perp$.
Output $\mathbb{I}\{i \leq \mathsf{PRF}_{\mathsf{Enc}}(c)\}$.

Figure 6: Modified program P^1. i^* and $x^* = \mathsf{PRG}(\mathsf{PRF}_{sk}(i^*))$ are hardcoded values.

input-output behavior of the programs are identical, thus the indistinguishability obfuscation guarantees that the obfuscations of these programs are computationally indistinguishable.

Next we modify the setup procedure to give a uniformly random value for s_{i^*}. The new setup procedure is indistinguishable from the original setup procedure by the pseudorandomness of $s_{i^*} = \mathsf{PRF}_{sk}(i^*)$. Finally, we modify the decryption program to use a truly random value x^* instead of $x^* = \mathsf{PRG}(\mathsf{PRF}_{sk}(i^*))$. The new decryption program is indistinguishable from the original by pseudorandomness of PRG and PRF_{sk}.

After making these modifications, with probability at least $1 - 2^{-\lambda/2}$, the random value x^* is not in the image of PRG. Thus, with probability at least $1 - 2^{-\lambda/2}$, the condition $\mathsf{PRG}(sk) = x^*$ will be unsatisfiable. Therefore, we can simply remove this test without changing the program on any inputs. Thus, the obfuscation of P^1 will be indistinguishable from the obfuscation of the following program P^2.

$\mathsf{P}^2_{\mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}, i^*}(c, i, s)$:
If $i = i^*$, halt and output $\perp$.
If $i \neq i^*$ and $\mathsf{PRG}(s) \neq \mathsf{PRG}(\mathsf{PRF}_{sk}^{\{i^*\}}(i))$, halt and output $\perp$.
Output $\mathbb{I}\{i \leq \mathsf{PRF}_{\mathsf{Enc}}(c)\}$.

Figure 7: Modified program P^2.

Breaking the decryption program for the challenge ciphertexts

First we modify the program so that the behavior on the challenge ciphertexts is hardcoded and $\mathsf{PRF}_{\mathsf{Enc}}$ is punctured on the challenge ciphertexts. The new decryption program is as follows. Note that the final line of the program is never reached when the input satisfies $c = c_0$ or $c = c_1$,

$\mathsf{P}^3_{\mathsf{PRF}_{sk}^{\{i^*\}},\mathsf{PRF}_{\mathsf{Enc}}^{\{c_0,c_1\}},i^*,c_0,b_0,c_1,b_1}(c,i,s)$:

If $i = i^*$, halt and output $\perp$.

If $i \neq i^*$ and $\mathsf{PRG}(s) \neq \mathsf{PRG}(\mathsf{PRF}_{sk}^{\{i^*\}}(i))$, halt and output $\perp$.

If $c = c_0$, output $\mathbb{I}\{i \leq i^* - b_0\}$

If $c = c_1$, output $\mathbb{I}\{i \leq i^* - b_1\}$

Output $\mathbb{I}\{i \leq \mathsf{PRF}_{\mathsf{Enc}}^{\{c_0,c_1\}}(c)\}$.

Figure 8: Modified program P^3. c_0, b_0, c_1, b_1 are hardcoded values.

so puncturing $\mathsf{PRF}_{\mathsf{Enc}}$ at these points does not affect the output of the program on any input. Thus, P^3 is indisintinguishable from P^2 by the security of indistinguishability obfuscation.

Next, since, $b_0, b_1 \in \{0,1\}$, and the decryption program halts immediately if $i = i^*$, the values of b_0, b_1 do not affect the output of the program. Thus, we can simply drop them from the description of the program without changing the program on any input. So, by security of the indistinguishability obfuscation, P^3 is indistinguishable from the following program P^4.

$\mathsf{P}^4_{\mathsf{PRF}_{sk}^{\{i^*\}},\mathsf{PRF}_{\mathsf{Enc}}^{\{c_0,c_1\}},i^*,c_0,c_1}(c,i,s)$:

If $i = i^*$, halt and output $\perp$.

If $i \neq i^*$ and $\mathsf{PRG}(s) \neq \mathsf{PRG}(\mathsf{PRF}_{sk}^{\{i^*\}}(i))$, halt and output $\perp$.

If $c = c_0$, output $\mathbb{I}\{i \leq i^*\}$

If $c = c_1$, output $\mathbb{I}\{i \leq i^*\}$

Output $\mathbb{I}\{i \leq \mathsf{PRF}_{\mathsf{Enc}}^{\{c_0,c_1\}}(c)\}$.

Figure 9: Modified program P^4. c_0, c_1 are hardcoded values.

Reducing to Input-Matching Security

Finally, we claim that if the adversary is able to win at **TwoIndexHiding** then he can also win the game **InputMatching**$[i^*-1, i^*]$, which violates input-matching security of $\mathcal{F}_{\lambda,\mathsf{Enc}}$.

Recall that the challenge in the game **InputMatching**$[i^*-1, i^*]$ consists of a tuple $(c_0, c_1, \mathsf{PRF}^{\{c_0,c_1\}})$ where $\mathsf{PRF}_{\mathsf{Enc}}$ is sampled subject to 1) $\mathsf{PRF}_{\mathsf{Enc}}(c_0) = i^* - b_0$ for a random $b_0 \in \{0,1\}$, 2) $\mathsf{PRF}_{\mathsf{Enc}}(c_1) = i^* - b_1$ for a random $b_1 \in \{0,1\}$, and 3) $\mathsf{PRF}_{\mathsf{Enc}}^{-1}(i) \neq \emptyset$ for every $i \in [n]$. Given this input, we can precisely simulate the view of the adversary in **TwoIndexHiding**$[i^*]$. To do so, we can choose PRF_{sk} and give the keys sk_{-i^*} and obfuscations of $\mathsf{P}^4_{\mathsf{PRF}_{sk}^{\{i^*\}},\mathsf{PRF}_{\mathsf{Enc}}^{\{c_0,c_1\}},i^*,c_0,c_1}$ to the adversary. Then we can user c_0, c_1 as the challenge ciphertexts and obtain a bit b' from the adversary. By input-matching security, we have that

$$\mathbb{P}\left[b' = b_0 \oplus b_1\right] - \frac{1}{2} = o(1/n^3).$$

Since, as we argued above, the view of the adversary in this game is indistinguishable from the view of the adversary in **TwoIndexHiding**$[i^*]$, we conclude that

$$\mathbb{P}_{\mathbf{TwoIndexHiding}[i^*]}\left[b' = b_0 \oplus b_1\right] - \frac{1}{2} = o(1/n^3),$$

as desired. This completes the proof.

7 A Weak Traitor-Tracing Scheme with Very Short Keys

In this section we construct a different traitor-tracing scheme for n users where the parameters are essentially reversed—the length of the secret user keys is $O(\log(n))$ and the length of the ciphertexts is $\text{poly}(\lambda)$. This scheme will be used to establish our hardness result for differential privacy when the number of queries is exponentially large but the data universe has only polynomial size.

7.1 Construction

Let $n = \text{poly}(\lambda)$ denote the number of users for the scheme. Let $m = \tilde{O}(n^6)$ be a parameter. Our construction will rely on the following primitives:

- A puncturable pseudorandom function family $\mathcal{F}_{\lambda,sk} = \{\mathsf{PRF}_{sk} : [n] \to [m]\}$.
- A puncturable pseudorandom function family $\mathcal{F}_{\lambda,\mathsf{Enc}} = \{\mathsf{PRF}_{\mathsf{Enc}} : [n] \times [m] \to \{0,1\}\}$.
- An iO scheme Obfuscate.

$\mathsf{Setup}(\lambda)$:
  Choose a pseudorandom function $\mathsf{PRF}_{sk} \leftarrow_R \mathcal{F}_{\lambda,sk}$.
  For $i = 1,\ldots,n$, let $s_i = \mathsf{PRF}_{sk}(i)$, and let each user's secret key be $sk_i = (i, s_i) \in [n] \times [m]$.
  Let the master key be $mk = \mathsf{PRF}_{sk}$.

$\mathsf{Enc}(j, mk = \mathsf{PRF}_{sk})$:
  Choose a pseudorandom function $\mathsf{PRF}_{\mathsf{Enc}} \leftarrow_R \mathcal{F}_{\lambda,\mathsf{Enc}}$.
  Let $\mathsf{O} = \mathsf{Obfuscate}(\mathsf{P}_{j,\mathsf{PRF}_{sk},\mathsf{PRF}_{\mathsf{Enc}}})$
  Output $c = \mathsf{O}$.

$\mathsf{Dec}(sk_i = (i, s_i), c = \mathsf{O})$:
  Output $\mathsf{O}(i, sk_i)$.

$\mathsf{P}_{j,\mathsf{PRF}_{sk},\mathsf{PRF}_{\mathsf{Enc}}}(i, s)$:
  If $s \neq \mathsf{PRF}_{sk}(i)$, output $\mathsf{PRF}_{\mathsf{Enc}}(i, s)$.
  Else, output $\mathbb{I}\{i \leq j\}$.

Figure 10: Our scheme $\Pi_{\text{short-key}}$

Theorem 7.1. *Assuming the existence of one-way functions and indistinguishability obfuscation, for every polynomial n, the scheme $\Pi_{\text{short-key}}$ is an (n, d, ℓ)-traitor-tracing scheme for $2^d = \tilde{O}(n^7)$ and $\ell = \text{poly}(\lambda)$, and is weakly index-hiding secure.*

Combining this theorem with Lemma 3.4 and Theorem 4.1 establishes Theorem 1.2 in the introduction.

Parameters

First we verify that $\Pi_{\text{short-key}}$ is an (n, d, ℓ)-traitor-tracing scheme for the desired parameters. Observe that the length of the secret keys is d such that $2^d = nm$. By construction, since $m = \tilde{O}(n^6)$, $2^d = \tilde{O}(n^7)$. The length of the ciphertext is $|\mathsf{O}|$, which is $\text{poly}(|\mathsf{P}|)$ by the efficiency of the obfuscation scheme. By the efficiency of the pseudorandom function family and the pairwise independent hash family, the running time of P is at most $\text{poly}(\lambda + \log(n))$. Since n is assumed to be a polynomial in λ, the ciphertexts have length $\text{poly}(\lambda)$.

7.2 Proof of Weak Index-Hiding Security

Just as in Section 6, we will rely on Lemma 3.4 so that we only need to show that for every $\lambda \in \mathbb{N}$, and every $i^* \in [n(\lambda)]$,

$$\mathop{\mathbb{P}}_{\textbf{TwoIndexHiding}[i^*]}\left[b' = b_0 \oplus b_1\right] - \frac{1}{2} = o(1/n^3).$$

We will demonstrate this using a series of hybrids to reduce security of the scheme in the **TwoIndexHiding** game to the security of the pseudorandom function families.

In our argument, recall that the adversary's view consists of the keys sk_{-i^*} and the challenge ciphertexts c_0, c_1. In our proof, we will not modify how the keys are generated, so we will present the hybrids only by how the challenge ciphertexts are generated. Also, for simplicity, we will focus only on how c_0 is generated as a function of i^*, b_0 and mk. The ciphertext c_1 will be generated in exactly the same way but as a function of i^*, b_1 and mk. We also remark that we highlight the changes in the hybrids in green.

Hiding the missing user key

First we modify the encryption procedure to one where PRF_{sk} is punctured on i^* and the value $s^* = \mathsf{PRF}_{sk}(i^*)$ is hardcoded into the program.

We claim that, by the security of the iO scheme, the distribution of c_0, c_1 under Enc^1 is computationally indistinguishable from the distribution of c_0, c_1 under Enc. The reason is that the obfuscation P and P^1 compute the same function. Consider two cases, depending on whether $i = i^*$ or $i \neq i^*$. If $i \neq i^*$, since $b_0 \in \{0, 1\}$, and $i \neq i^*$, replacing $\mathbb{I}\{i \leq i^* - b_0\}$ with $\mathbb{I}\{i \leq i^* - 1\}$ does not change the output. Moreover, since we only reach the branch involving $\mathsf{PRF}_{sk}^{\{i^*\}}$ when $i \neq i^*$, the puncturing does not affect the output of the program. If $i = i^*$, then the program either outputs $\mathsf{PRF}_{\mathsf{Enc}}(i^*, s)$ as it did before when $s \neq s^*$ or it outputs $1 - b_0$: equivalent to $\mathbb{I}\{i \leq i^* - b_0\}$. Thus, by iO, the obfuscated programs are indistinguishable.

Next, we argue that, since $\mathsf{PRF}_{sk}^{\{i^*\}}$ is sampled from a puncturable pseudorandom function family, and the adversary's view consists of $s_{-i^*} = \{\mathsf{PRF}_{sk}(i)\}_{i \neq i^*}$ but not $\mathsf{PRF}_{sk}(i^*)$, the value of $\mathsf{PRF}_{sk}(i^*)$ is computationally indistinguishable to the adversary from a random value. Thus, we can move to another hybrid $(\mathsf{Enc}^2, \mathsf{P}^2)$ where the value s^* is replaced with a uniformly random value $\tilde{s}$.

Hiding the challenge index Now we want to remove any explicit use of b_0 from P^2. The natural way to try to do this is to remove the line where the program outputs $1 - b_0$ when the input is $(i^*, \tilde{s})$, and instead have the program output $\mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s})$. However, this would involve changing the program's output on one input, and indistinguishability obfuscation does not guarantee any security in this case. We get around this problem in two steps. First, we note that the value of

$\mathsf{Enc}^1(i^*, b_0, mk = \mathsf{PRF}_{sk})$:

Choose a pseudorandom function $\mathsf{PRF}_{\mathsf{Enc}} \leftarrow_{\mathrm{R}} \mathcal{F}_{\lambda,\mathsf{Enc}}$.

Let $s^* = \mathsf{PRF}_{sk}(i^*)$, $\mathsf{PRF}_{sk}^{\{i^*\}} = \mathsf{Puncture}(\mathsf{PRF}_{sk}, i^*)$.

$$\text{Let } \mathsf{O} = \mathsf{Obfuscate}\left(\mathsf{P}^1_{i^*, b_0, s^*, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}}\right).$$

Output $c_0 = \mathsf{O}$.

$\mathsf{P}^1_{i^*, b_0, s^*, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}}(i, s)$:

If $i = i^*$

If $s \neq s^*$, output $\mathsf{PRF}_{\mathsf{Enc}}(i^*, s)$

If $s = s^*$, output $1 - b_0$

Else If $i \neq i^*$

If $s \neq \mathsf{PRF}_{sk}^{\{i^*\}}(i)$, halt and output $\mathsf{PRF}_{\mathsf{Enc}}(i, s)$.

Output $\mathbb{I}\{i \leq i^* - 1\}$.

Figure 11: Hybrid $(\mathsf{Enc}^1, \mathsf{P}^1)$.

$\mathsf{Enc}^2(i^*, b_0, mk = \mathsf{PRF}_{sk})$:

Choose a pseudorandom function $\mathsf{PRF}_{\mathsf{Enc}} \leftarrow_{\mathrm{R}} \mathcal{F}_{\lambda,\mathsf{Enc}}$.

$\mathsf{PRF}_{sk}^{\{i^*\}} = \mathsf{Puncture}(\mathsf{PRF}_{sk}, i^*)$, Let $\tilde{s} \leftarrow_{\mathrm{R}} [m]$.

$$\text{Let } \mathsf{O} = \mathsf{Obfuscate}\left(\mathsf{P}^2_{i^*, b_0, \tilde{s}, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}}\right).$$

Output $c_0 = \mathsf{O}$.

$\mathsf{P}^2_{i^*, b_0, \tilde{s}, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}}(i, s)$:

If $i = i^*$

If $s \neq \tilde{s}$, output $\mathsf{PRF}_{\mathsf{Enc}}(i^*, s)$

If $s = \tilde{s}$, output $1 - b_0$

Else If $i \neq i^*$

If $s \neq \mathsf{PRF}_{sk}^{\{i^*\}}(i)$, halt and output $\mathsf{PRF}_{\mathsf{Enc}}(i, s)$.

Output $\mathbb{I}\{i \leq i^* - 1\}$.

Figure 12: Hybrid $(\mathsf{Enc}^2, \mathsf{P}^2)$.

$\mathsf{PRF}_{\mathsf{Enc}}$ on the point $(i^*, \tilde{s})$ is never needed in P^2, so we can move to a new procedure P^3 where we puncture at that point without changing the program functionality. Indistinguishability obfuscation guarantees that P^2 and P^3 are computationally indistinguishable.

Next, we define another hybrid P^4 where change how we sample $\mathsf{PRF}_{\mathsf{Enc}}$ and sample it so that $\mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s}) = 1 - b_0$. Observe that the hybrid only depends on $\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*, \tilde{s})\}}$. We claim the distributions of $\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*, \tilde{s})\}}$ when $\mathsf{PRF}_{\mathsf{Enc}}$ is sampled correctly versus sampled conditioned on

$\mathsf{Enc}^3(i^*, b_0, mk = \mathsf{PRF}_{sk})$:

Let $\tilde{s} \leftarrow_R [m]$.

Choose a pseudorandom function $\mathsf{PRF}_{\mathsf{Enc}} \leftarrow_R \mathcal{F}_{\lambda,\mathsf{Enc}}$

$\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}} = \mathsf{PuncturePRF}_{\mathsf{Enc}}, (i^*, \tilde{s})$.

$\mathsf{PRF}_{sk}^{\{i^*\}} = \mathsf{Puncture}(\mathsf{PRF}_{sk}, i^*)$.

$$\text{Let } \mathsf{O} = \mathsf{Obfuscate}\left(\mathsf{P}^3_{i^*,b_0,\tilde{s},\mathsf{PRF}_{sk}^{\{i^*\}},\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}}\right).$$

Output $c_0 = \mathsf{O}$.

$\mathsf{P}^3_{i^*,b_0,\tilde{s},\mathsf{PRF}_{sk}^{\{i^*\}},\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}}(i, s)$:

If $i = i^*$

If $s \neq \tilde{s}$, output $\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}(i^*, s)$

If $s = \tilde{s}$, output $1 - b_0$

Else If $i \neq i^*$

If $s \neq \mathsf{PRF}_{sk}^{\{i^*\}}(i)$, halt and output $\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}(i, s)$.

Output $\mathbb{I}\{i \leq i^* - 1\}$.

Figure 13: Hybrid $(\mathsf{Enc}^3, \mathsf{P}^3)$.

$\mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s}) = 1 - b_0$ are computationally indistinguishable. This follows readily from punctured PRF security. Suppose to the contrary that the two distributions were distinguishable with non-negligible advantage δ by adversary A. Then consider a punctured PRF adversary B that is given $\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}, b$ where b is chosen at random, or $b = \mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s})$. B distinguishes the two cases as follows. If $b \neq 1 - b_0$, then B outputs a random bit and stops. Otherwise, it runs A on $\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}$, and outputs whatever A outputs. If b is truly random and independent of $\mathsf{PRF}_{\mathsf{Enc}}$, then conditioned on $b = 1 - b_0$, $\mathsf{PRF}_{\mathsf{Enc}}$ is sampled randomly. However, if $b = \mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s})$, then conditioned on $b = 1 - b_0$, $\mathsf{PRF}_{\mathsf{Enc}}$ is sampled such that $\mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s}) = 1 - b_0$. These are exactly the two cases that A distinguishes. Hence, conditioned on $b = 1 - b_0$, B guesses correctly with probability $\frac{1}{2} + \delta$. Moreover, by PRF security, $b = 1 - b_0$ with probability $\geq \frac{1}{2} - \varepsilon$ for some negligible quantity ε, and in the case $b \neq 1 - b_0$, B guess correctly with probability $\frac{1}{2}$. Hence, overall B guesses correctly with probability $\geq \frac{1}{2}(\frac{1}{2} + \varepsilon) + (\frac{1}{2} + \delta)(\frac{1}{2} - \varepsilon) = \frac{1}{2} + \frac{\delta}{2} - \varepsilon\delta$. Hence, B has non-negligible advantage $\frac{\delta}{2} - \varepsilon\delta$. Thus, changing how $\mathsf{PRF}_{\mathsf{Enc}}$ is sampled is computationally undetectable, and P is otherwise unchanged. Therefore P^3 and P^4 are computationally indistinguishable.

Next, since $\mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s}) = 1 - b_0$, we can move to another hybrid P^5 where we delete the line "If $s = \tilde{s}$, output $1 - b_0$" without changing the functionality. Thus, by indistinguishability obfuscation, P^4 and P^5 are computationally indistinguishable.

Now notice that P^5 is independent of b_0. However, Enc^5 still depends on b_0. We now move to the final hybrid P^6 where we remove the condition that $\mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s}) = 1 - b_0$, which will completely remove the dependence on b_0.

To prove that Enc^6 is indistinguishable from Enc^5, notice that they are independent of $\tilde{s}$, except through the sampling of $\mathsf{PRF}_{\mathsf{Enc}}$. Using this, and the following lemma, we argue that we can remove the condition that $\mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s}) = 1 - b_0$.

$\mathsf{Enc}^4(i^*, b_0, mk = \mathsf{PRF}_{sk})$:

Let $\tilde{s} \leftarrow_R [m]$.

Choose a pseudorandom function $\mathsf{PRF}_{\mathsf{Enc}} \leftarrow_R \mathcal{F}_{\lambda,\mathsf{Enc}}$ conditioned on $\mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s}) = 1 - b_0$.

$\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}} = \mathsf{PuncturePRF}_{\mathsf{Enc}},(i^*, \tilde{s})$.

$\mathsf{PRF}_{sk}^{\{i^*\}} = \mathsf{Puncture}(\mathsf{PRF}_{sk}, i^*)$.

$$\text{Let } \mathsf{O} = \mathsf{Obfuscate}\left(\mathsf{P}^4_{i^*, b_0, \tilde{s}, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}}\right).$$

Output $c_0 = \mathsf{O}$.

$\mathsf{P}^4_{i^*, b_0, \tilde{s}, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}}(i, s)$:

If $i = i^*$

If $s \neq \tilde{s}$, output $\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}(i^*, s)$

If $s = \tilde{s}$, output $1 - b_0$

Else If $i \neq i^*$

If $s \neq \mathsf{PRF}_{sk}^{\{i^*\}}(i)$, halt and output $\mathsf{PRF}_{\mathsf{Enc}}^{\{(i^*,\tilde{s})\}}(i, s)$.

Output $\mathbb{I}\{i \leq i^* - 1\}$.

Figure 14: Hybrid $(\mathsf{Enc}^4, \mathsf{P}^4)$.

$\mathsf{Enc}^5(i^*, b_0, mk = \mathsf{PRF}_{sk})$:

Let $\tilde{s} \leftarrow_R [m]$.

Choose a pseudorandom function $\mathsf{PRF}_{\mathsf{Enc}} \leftarrow_R \mathcal{F}_{\lambda,\mathsf{Enc}}$ such that $\mathsf{PRF}_{\mathsf{Enc}}(i^*, \tilde{s}) = 1 - b_0$

$\mathsf{PRF}_{sk}^{\{i^*\}} = \mathsf{Puncture}(\mathsf{PRF}_{sk}, i^*)$.

$$\text{Let } \mathsf{O} = \mathsf{Obfuscate}\left(\mathsf{P}^5_{i^*, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}}\right).$$

Output $c_0 = \mathsf{O}$.

$\mathsf{P}^5_{i^*, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}}(i, s)$:

If $i = i^*$

Output $\mathsf{PRF}_{\mathsf{Enc}}(i^*, s)$

Else If $i \neq i^*$

If $s \neq \mathsf{PRF}_{sk}^{\{i^*\}}(i)$, halt and output $\mathsf{PRF}_{\mathsf{Enc}}(i, s)$.

Output $\mathbb{I}\{i \leq i^* - 1\}$.

Figure 15: Hybrid $(\mathsf{Enc}^5, \mathsf{P}^5)$.

Lemma 7.2. *Let $\mathcal{H} = \{h : [T] \to [K]\}$ be a δ-almost pairwise independent hash family. Let $y \in [K]$ and $M \subseteq [T]$ of size m be arbitrary. Define the following two distributions.*

- *D_1: Choose $h \leftarrow_R \mathcal{H}$.*

$\mathsf{Enc}^6(i^*, mk = \mathsf{PRF}_{sk})$:

Choose a pseudorandom function $\mathsf{PRF}_{\mathsf{Enc}} \leftarrow_R \mathcal{F}_{\lambda,\mathsf{Enc}}$

$\mathsf{PRF}_{sk}^{\{i^*\}} = \mathsf{Puncture}(\mathsf{PRF}_{sk}, i^*)$.

$$\text{Let } \mathsf{O} = \mathsf{Obfuscate}\left(\mathsf{P}^6_{i^*, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}}\right).$$

Output $c_0 = \mathsf{O}$.

$\mathsf{P}^6_{i^*, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc}}}(i, s)$:

If $i = i^*$

Output $\mathsf{PRF}_{\mathsf{Enc}}(i^*, s)$

Else If $i \neq i^*$

If $s \neq \mathsf{PRF}_{sk}^{\{i^*\}}(i)$, halt and output $\mathsf{PRF}_{\mathsf{Enc}}(i, s)$.

Output $\mathbb{I}\{i \leq i^* - 1\}$.

Figure 16: Hybrid $(\mathsf{Enc}^6, \mathsf{P}^6)$.

- *D_2: Choose a random $x \in M$, and then choose $h \leftarrow_R (\mathcal{H} \mid h(x) = y)$.*

Then D_1 and D_2 are $(\frac{1}{2}\sqrt{K/m + 7K^2\delta})$-close in statistical distance.

We defer the proof to Section 7.3. The natural way to try to show that $(\mathsf{Enc}^6, \mathsf{P}^6)$ is $o(1/n^3)$ statistically close to $(\mathsf{Enc}^5, \mathsf{P}^5)$ is to apply this lemma to the hash family $\mathcal{H} = \mathcal{F}_{\lambda,\mathsf{Enc}}$. Recall that a pseudorandom function family is also $\mathrm{negl}(\lambda)$-pairwise independent. Here, the parameters would be $[T] = [n] \times [m]$, $M = \{(i^*, s) \mid s \in [m]\}$ and $b = 1 - b_0$, and the random choice $x \in M$ is the pair $(i^*, \tilde{s})$.

However, recall that the adversary not only sees $c_0 = \mathsf{Enc}^5(i^*, b_0, mk)$, but also sees $c_1 = \mathsf{Enc}^5(i^*, b_1, mk)$, and these share the same $\tilde{s}$. Hence, we cannot directly invoke Lemma 7.2 on the $\mathsf{PRF}_{\mathsf{Enc},0}$ sampled in c_0, since $\tilde{s}$ is also used to sample $\mathsf{PRF}_{\mathsf{Enc},1}$ when sampling c_1, and is therefore not guaranteed to be random given c_1.

Instead, we actually consider the function family $\mathcal{H} = \mathcal{F}^2_{\lambda,\mathsf{Enc}}$, where we define

$$h(i, s) = (\mathsf{PRF}_{\mathsf{Enc},0}, \mathsf{PRF}_{\mathsf{Enc},1})(i, s) = (\mathsf{PRF}_{\mathsf{Enc},0}(i, s), \mathsf{PRF}_{\mathsf{Enc},1}(i, s)).$$

In Enc^5, h is drawn at random conditioned on $h(i^*, \tilde{s}) = (1 - b_0, 1 - b_1)$, whereas in Enc^6, it is drawn at random.

$\mathcal{H}$ is still a pseudorandom function family, so it must be $\mathrm{negl}(\lambda)$-almost pairwise independent with δ negligible. In particular, $\delta = o(1/m)$. Hence, the conditions of Lemma 7.2 are satisfied with $K = 4$. Since the description of $\mathsf{P}^5, \mathsf{P}^6$ is the tuple $(i^*, \tilde{s}, \mathsf{PRF}_{sk}^{\{i^*\}}, \mathsf{PRF}_{\mathsf{Enc},0}, \mathsf{PRF}_{\mathsf{Enc},1})$, and by Lemma 7.2 the distribution on these tuples differs by at most $O(\sqrt{1/m})$ in statistical distance, we also have that the distribution on obfuscations of $\mathsf{P}^5, \mathsf{P}^6$ differs by at most $O(\sqrt{1/m})$. Finally, we can choose a value of $m = \tilde{O}(n^6)$ so that $O(\sqrt{1/m}) = o(1/n^3)$.

Observe that when we generate user keys sk_{-i^*} and the challenge ciphertexts according to $(\mathsf{Enc}^6, \mathsf{P}^6)$, the distribution of the adversary's view is completely independent of the random values b_0, b_1. Thus no adversary can output $b' = b_0 \oplus b_1$ with probability greater than 1/2. Since

the distribution of these challenge ciphertexts is $o(1/n^3)$-computationally indistinguishable from the original distribution on challenge ciphertexts, we have that for every computationally efficient adversary,

$$\mathop{\mathbb{P}}_{\mathbf{TwoIndexHiding}[i^*]}\left[b' = b_0 \oplus b_1\right] - \frac{1}{2} = o(1/n^3),$$

as desired. This completes the proof.

7.3 Proof of Lemma 7.2

We will fix $y = 1$ for simplicity. The cases of $y = 2, \ldots, K$ follow symmetrically.

We will first bound the Rényi divergence between D_1 and D_2, which is defined as

$$RD(D_1, D_2) = \sum_h \frac{\mathbb{P}[H = h : H \leftarrow_{\mathrm{R}} D_2]^2}{\mathbb{P}[H = h : H \leftarrow_{\mathrm{R}} D_1]}$$

Here, h ranges over the support of D_2 (since the support of D_2 is a subset of H, we can equivalently view the sum as one over all h in H). Once we do this, we will obtain an upper bound on the statistical distance between D_1 and D_2 using the inequality

$$SD(D_1, D_2) \leq \frac{\sqrt{RD(D_1, D_2) - 1}}{2}. \tag{9}$$

To bound the Rényi divergence, we can start by writing

$$\begin{aligned}\mathbb{P}[H = h : H \leftarrow D_2]^2 &= \left(\frac{1}{m}\sum_{x \in M} \mathbb{P}[H = h : H(x) = 1]\right)^2 \\ &= \frac{1}{m^2}\sum_{x, x' \in M} \mathbb{P}[H = h : H(x) = 1]\,\mathbb{P}[H = h : H(x') = 1]\end{aligned}$$

Where in all the (conditional) probabilities on the right, h is drawn from D_1, conditioned on some event. This allows us to write

$$RD(D_1, D_2) = \frac{1}{m^2}\sum_{x, x' \in M}\sum_h \frac{\mathbb{P}[H = h : H(x) = 1]\,\mathbb{P}[H = h : H(x') = 1]}{\mathbb{P}[H = h]}$$

We now divide the sum into two cases.

- $x = x'$. In this case, the summand becomes $\mathbb{P}[H = h : H(x) = 1]^2 / \mathbb{P}[H = h]$. Notice that

  $$\mathbb{P}[H = h : H(x) = 1] = \begin{cases} 0 & \text{if } h(x) \neq 1 \\ \frac{\mathbb{P}[H=h]}{\mathbb{P}[H(x)=1]} & \text{if } h(x) = 1 \end{cases}$$

  Therefore, the summand is

  $$\frac{\mathbb{P}[H = h : H(x) = 1]^2}{\mathbb{P}[H = h]} = \begin{cases} 0 & \text{if } h(x) \neq 1 \\ \frac{\mathbb{P}[H=h]}{\mathbb{P}[H(x)=1]^2} & \text{if } h(x) = 1 \end{cases}$$

  Now, notice that $\sum_{h:h(x)=1} \mathbb{P}[H = h] = \mathbb{P}[H(x) = 1]$. Thus, if we carry out the sum over h, the summand becomes $1/\mathbb{P}[H(x) = 1]$.

- $x \neq x'$. Then

$$\begin{aligned}\mathbb{P}[H = h : H(x) = 1] = \mathbb{P}[H = h : H(x) = 1, H(x') \neq 1]\mathbb{P}[H(x') \neq 1 : H(x) = 1] + \\ \mathbb{P}[H = h : H(x) = 1, H(x') = 1]\mathbb{P}[H(x') = 1 : H(x) = 1] \\ \mathbb{P}[H = h : H(x') = 1] = \mathbb{P}[H = h : H(x') = 1, H(x) \neq 1]\mathbb{P}[H(x) \neq 1 : H(x') = 1] + \\ \mathbb{P}[H = h : H(x') = 1, H(x) = 1]\mathbb{P}[H(x) = 1 : H(x') = 1]\end{aligned}$$

  When we take the product of the two expressions and expand, we obtain four products, only one of which is nonzero (when $h(x) = h(x') = 1$):

$$\mathbb{P}[H = h : H(x) = 1, H(x') = 1]^2 \cdot \mathbb{P}[H(x') = 1 : H(x) = 1] \cdot \mathbb{P}[H(x) = 1 : H(x') = 1]$$

  Therefore, the summand is

$$\begin{aligned}&\frac{\mathbb{P}[H = h : H(x) = 1]\mathbb{P}[H = h : H(x') = 1]}{\mathbb{P}[H = h]} \\ &= \frac{\mathbb{P}[H(x') = 1 : H(x) = 1]\mathbb{P}[H(x) = 1 : H(x') = 1]}{\mathbb{P}[H = h]} \cdot \mathbb{P}[H = h : H(x) = 1, H(x') = 1]^2 \\ &= \frac{\mathbb{P}[H = h]}{\mathbb{P}[H(x) = 1]\mathbb{P}[H(x') = 1]} \cdot \mathbb{I}\{h(x) = h(x') = 1\}\end{aligned}$$

  When we sum over all h, we get

$$\sum_h \frac{\mathbb{P}[H = h : H(x) = 1]\mathbb{P}[H = h : H(x') = 1]}{\mathbb{P}[H = h]} = \frac{\mathbb{P}[H(x) = 1 \wedge H(x') = 1]}{\mathbb{P}[H(x) = 1]\mathbb{P}[H(x') = 1]}$$

  Therefore, the Rényi divergence is

$$RD(D_1, D_2) = \frac{1}{m^2}\left(\left(\sum_{x \in M} \frac{1}{\mathbb{P}[H(x) = 1]}\right) + \left(\sum_{x \neq x' \in M} \frac{\mathbb{P}[H(x) = 1 \wedge H(x') = 1]}{\mathbb{P}[H(x) = 1]\mathbb{P}[H(x') = 1]}\right)\right)$$

  We now invoke δ-almost pairwise independence to claim that

  - For every $x \in M$, $\mathbb{P}[H(x) = 1] \geq 1/K - \delta$.
  - For every $x \neq x' \in M$, $\mathbb{P}[H(x) = 1 \wedge H(x') = 1] \leq 1/K^2 + \delta$.

  Therefore, for $\delta \leq 1/2K$, it is easy to show that $1/\mathbb{P}[H(x) = 1] \leq 2 + 2K^2\delta$ and

$$\frac{\mathbb{P}[H(x) = 1 \wedge H(x') = 1]}{\mathbb{P}[H(x) = 1]\mathbb{P}[H(x') = 1]} \leq 1 + 7K^2\delta.$$

So we have that

$$RD(D_1, D_2) \leq \frac{1}{m^2}\left((K + 2K^2\delta)m + (m^2 - m)(1 + 7K^2\delta)\right) \leq 1 + \frac{K-1}{m} + 7K^2\delta.$$

Using the relationship between statistical distance and Rényi divergence above (Equation (9)), we obtain $SD(D_1, D_2) \leq \frac{1}{2}\sqrt{(K-1)/m + 7K^2\delta}$, as long as $\delta < 1/2k$. Notice that for $\delta \geq 1/2K$, $7K^2\delta \geq 7$, and so our bound is larger than 1 anyway. Hence, the bound holds for all δ.

Acknowledgments

We thank Dan Boneh for helpful discussions in the early stages of this work.

The first author is supported by an NSF Graduate Research Fellowship #DGE-11-44155. 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. Part of this work was done while the third author was a postdoctoral fellow in the Columbia University Department of Computer Science, supported by a junior fellowship from the Simons Society of Fellows. 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.

References

[BDMN05] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *PODS*, 2005.

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and uces: The case of computationally unpredictable sources. In *CRYPTO*, 2014.

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

[BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In *EUROCRYPT*, 2016.

[BNS13] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In *RANDOM*, 2013.

[BNSV15] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, 2015.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos. In *TCC*, 2016.

[BST16] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Contention in cryptoland: Obfuscation, leakage and uce. In *TCC*, 2016.

[BUV14] Mark Bun, Jonathan Ullman, and Salil P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. In *STOC*, 2014.

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

[BZ16] Mark Bun and Mark Zhandry. Order-revealing encryption and the hardness of private learning. In *TCC*, 2016.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, pages 257–270, 1994.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New mmap attacks and their limitations. In *CRYPTO*, 2015.

[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'14, Princeton, NJ, USA, January 12-14, 2014*, pages 387–402, 2014.

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

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, 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 *STOC*, 2009.

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

[DRV10] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*. IEEE, 2010.

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

[GGH+13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 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.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In *FOCS*, 2015.

[GMS16] Sanjam Garg, Pratyay Mukherjee, and Akshayaram Srinivasan. Obfuscation without the vulnerabilities of multilinear maps. Cryptology ePrint Archive, Report 2016/390, 2016. http://eprint.iacr.org/.

[GRU12] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *TCC*, 2012.

[HR10] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, 2010.

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

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In *EUROCRYPT*, 2014.

[HU14] Moritz Hardt and Jonathan Ullman. Preventing false discovery in interactive data analysis is hard. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 454–463, 2014.

[Kea98] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6), 1998.

[KLN+11] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. In *CRYPTO*, 2016.

[NTZ13] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *STOC*, 2013.

[RR10] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *STOC*, pages 765–774. ACM, June 5–8 2010.

[SU15a] Thomas Steinke and Jonathan Ullman. Between pure and approximate differential privacy. *CoRR*, abs/1501.06095, 2015.

[SU15b] 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 2015, Paris, France, July 3-6, 2015*, pages 1588–1628, 2015.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *STOC*, 2014.

[TUV12] Justin Thaler, Jonathan Ullman, and Salil P. Vadhan. Faster algorithms for privately releasing marginals. In *ICALP*, 2012.

[Ull13] Jonathan Ullman. Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. In *STOC*, 2013.

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

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

A Twice Puncturable Pseudorandom Functions

A.1 An Input-Matching Secure PRF

Like pseudorandom functions satisfying the existing notions of puncturing, our construction is simply the GGM PRF family. We detail this constraint for notational purposes. For simplicity, we assume that m and n are powers of 2. We now claim that the GGM construction satisfies

PRF :
Parameters: a seed $s \in \{0,1\}^\lambda$, a pseudorandom generator $PRG : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$
and domain and range sizes $n = \text{poly}(\lambda), m = \tilde{O}(n^7)$
Input: $x \in [m] = \{0,1\}^\mu$ where $\mu = \log_2(m)$
Let $z = PRG_{x_\mu}(PRG_{x_{\mu-1}}(\ldots(PRG_{x_2}(PRG_{x_1}(s)))\ldots))$
Let y be the first $\log_2(n)$ bits of z and output y

Setup(1^λ):
Draw $s : \{0,1\}^\lambda$ at random conditioned on every $y \in [n]$ having a preimage under the PRF defined by s.
Note that this is efficiently computable since $m = \omega(n \lg n)$ by the reasoning in section 5.3.

Puncture(PRF, x_0, x_1):
Note that the pseudorandom function PRF defines a tree of seed values
(one for each node of the tree - the root is s, the left child is $PRG(s)_0$, etc)
Output $\text{PRF}^{\{x_0,x_1\}}$ = the set of all seed values of each node which is not an ancestor of x_0 or x_1 but its parent is. (note there are $O(\lg m)$ such seeds)

$\text{PRF}^{\{x_0,x_1\}}$
Input: $x \in [m] = \{0,1\}^\mu$ where $\mu = \log_2(m)$
Let s_j be the seed of the first node which is an ancestor of x and not an ancestor of x_0, x_1 but it parent is. Let j be its height in the binary tree.
Let $z = PRG_{x_\mu}(PRG_{x_{\mu-1}}(\ldots(PRG_{x_{j+1}}(PRG_{x_j}(s_j)))\ldots))$
Let y be the first $\log_2(n)$ bits of z and output y

Figure 17: The GGM pseudorandom function family $\{\text{PRF} : [m] \to [n]\}$

Theorem 5.4.

Theorem A.1. *If m, n are polynomial in the security parameter and $m = \omega(n \log(n))$, then the GGM pseudorandom function is ε-input-matching secure for $\varepsilon = \tilde{O}(\sqrt{n/m})$.*

We start by modifying the InputMatching game to one that will make it easier to prove security. Consider the following pair of games.

- Game0: The InputMatching$[y_0, y_1]$ game above.
- Game1: We modify the InputMatching$[y_0, y_1]$ game in the following way. Instead of choosing PRF conditioned on $\forall y \in [n]$, $\text{PRF}^{-1}(y) \neq \emptyset$, we first choose $x_0, x_1 \leftarrow_R [m]$ and then choose

PRF conditioned on $\mathsf{PRF}(x_0) = y_{b_0}$ and $\mathsf{PRF}(x_1) = y_{b_1}$. We will prove that the challenges $(x_0, x_1, \mathsf{PRF}^{\{x_0,x_1\}})$ in Game 0 and Game1 are statistically indistinguishable with suitable parameters.

Claim A.2. *For every polynomials m, n,* Game0 *and* Game1 *are ε-computationally indistinguishable for $\varepsilon = \tilde{O}(\sqrt{n/m})$.*

The proof is an uninsightful computation, so we will defer it to Section A.1.1. Now we want to prove that the $b_0 \oplus b_1 = 0$ and $b_0 \oplus b_1 = 1$ cases of Game1 are computationally indistinguishable.

Claim A.3.

$$\mathop{\mathbb{P}}_{\mathsf{Game1}}[b' = b_0 \oplus b_1] \leq \frac{1}{2} + O\left(\frac{1}{m}\right).$$

To show this, we need the following lemma:

Lemma A.4. *Let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG, and let $p : \{0,1\}^{2\lambda} \to \{0,1\}$ be an efficiently computable predicate on $\{0,1\}^{2\lambda}$. Suppose that, for a random $z \in \{0,1\}^{2\lambda}$, $\Pr[p(z) = 1]$ is non-negligible. Then the following distributions are ε-computationally indistinguishable for $\varepsilon = \mathrm{negl}(\lambda)$.*

- *z for $z = \mathsf{PRG}(s)$,where s is chosen at random from $\{0,1\}^\lambda$ conditioned on $p(\mathsf{PRG}(s)) = 1$.*
- *z where z is chosen at random from $\{0,1\}^{2\lambda}$ conditioned on $p(z) = 1$.*

Proof sketch. Given an efficient distinguisher D for the two distributions above, we construct the following efficient PRG distinguisher D'. D', on input z, computes $p(z)$. Since p and D are computationally efficient, so is D'. If the output is 0, then D' outputs a random bit. Otherwise, D' runs D on z and outputs the result. In the case where $p(z) = 0$, D' has no advantage, and in the case where $p(z) = 1$, D' has non-negligible advantage (namely the advantage of D). Since $p(z) = 1$ with non-negligible probability, D' has overall non-negligible advantage. □

Proof sketch of Claim A.3. Consider the GGM tree, and the punctured function $\mathsf{PRF}^{\{x_0,x_1\}}$ consisting of the values at all nodes in the tree that are not an ancestor of x_0 or x_1, but whose parent is an ancestor. We now consider the following procedure

1. Pick a node whose value s is random (perhaps conditioned on some predicate p on $\mathsf{PRG}(s)$), and not derived from another node's value. For example, at the beginning this is the root node, which is random, conditioned on the leaves at x_0 and x_1 having values y_{b_0}, y_{b_1}.
2. If that node is part of the punctured key $\mathsf{PRF}^{\{x_0,x_1\}}$ or is one of the leaves at x_0, x_1, don't do anything to this node.
3. Otherwise, delete that node, and replace the values of the children with random, conditioned on the predicate p being 1. This predicate is applied to the concatenation of the children's values. Notice that, given the form of our initial predicate each of the children's new values s_0, s_1 will be independently random, perhaps conditioned on some predicates p_0, p_1 on $\mathsf{PRG}(s_0), \mathsf{PRG}(s_1)$ respectively.

We iterate the procedure until we no longer make any changes to the tree. By applying Lemma A.4 to the change made in step 3, we can see that the distributions on the punctured PRF before and after we apply the procedure are ε-computationally indistinguishable for $\varepsilon = \mathrm{negl}(\lambda)$. In the end, we will have changed all of the punctured key values to uniformly random. Note

that, since these nodes are not ancestors of x_0, x_1, the predicate on them is trivially satisfied, so they are uniformly random even conditioned on the values of the function at x_0, x_1. Thus, these values are also independent of the output of the function at the points x_0, x_1. Thus, as long as $x_0 \neq x_1$ we can swap these values to any combination of y_{b_0} and y_{b_1}. Since the probability that $x_0 = x_1$ is at most $1/m$, the probability that the adversary can guess $b_0 \oplus b_1$ is at most $1/2 + 1/m$. Since this distribution is ε-computationally indistinguishable from the real distribution on the punctured PRF, a computationally efficient adversary can guess $b_0 \oplus b_1$ with probability at most $1/2 + 1/m + \mathrm{negl}(\lambda) = 1/2 + O(1/m)$. □

A.1.1 Proof of Claim A.2

Consider two ways of sampling a tuple $(y_0, y_1, b_0, b_1, x_0, x_1, \mathsf{PRF})$. Here $y_0, y_1 \in [n]$ are fixed, $b_0, b_1 \in \{0,1\}$ are uniformly random and independent. In Game0, PRF is sampled conditioned on every $y \in [n]$ having non-empty preimage, and x_0 and x_1 are random preimages of y_{b_0} and y_{b_1}, respectively. In Game1, $x_0, x_1 \in [m]$ are uniformly random and independent and PRF is sampled conditioned on $\mathsf{PRF}(x_0) = y_{b_0}$ and $\mathsf{PRF}(x_1) = y_{b_1}$. Observe that these tuples contain strictly more information than the challenges given to the adversary in Game0 and Game1, respectively. That is, the challenges can be generated by applying a function to these tuples, which cannot increase the statistical distance between the two distributions. Thus, to prove Claim A.2 it suffices to prove that these two distributions are statistically close.

First, we switch to an intermediate game Game0A in which the function PRF is *not* required to have a preimage for every $y \in [n]$. To make the sampling procedure well defined, if y_{b_0} or y_{b_1} does not have a preimage under PRF, we simply choose x_0 and x_1 at random from $[m]$.

Lemma A.5. *If* PRF *is pseudorandom and m, n are polynomials, then* Game0 *and* Game0A *are ε-statistically indistinguishable for $\varepsilon = n \cdot \exp(-\Omega(m/n)) + \mathrm{negl}(\lambda)$.*

Proof. Suppose that $f : [m] \to [n]$ is a uniformly random function. Then a simple calculation shows that the probability that there exists $y \in [n]$ such that $\mathsf{PRF}^{-1}(y) = \emptyset$ is at most

$$n \cdot (1 - 1/n)^m = n \cdot \exp(-\Omega(m/n)).$$

Since m, n are polynomial in the security parameter λ, there is a polynomial time algorithm that checks whether a function $\mathsf{PRF} : [m] \to [n]$ has at least one preimage for every $y \in [n]$. Thus, if PRF is sampled from a pseudorandom function family, it must also be true that the probability that there exists $y \in [n]$ such that $\mathsf{PRF}^{-1}(y) = \emptyset$ is at most $n \cdot \exp(-\Omega(m/n)) + \mathrm{negl}(\lambda)$, or else there would be an efficient algorithm that distinguishes a random function PRF from a truly random function f.

Since conditioning on an event that occurs with probability at least $1 - p$ can only affect the distribution by at most p in statistical distance, we conclude that the two distributions are statistically indistinguishable to within $n \cdot \exp(-\Omega(m/n)) + \mathrm{negl}(\lambda)$. □

Now, we introduce a second intermediate game Game0B in which we first choose a random value $x_0 \in [m]$, then sample PRF such that $\mathsf{PRF}(x_0) = y_{b_0}$, and finally we choose x_1 to be a random preimage of y_{b_1}. If y_{b_1} has no preimage under PRF, we choose x_1 at random from $[m]$.

Lemma A.6. *If* PRF *is pseudorandom and m, n are polynomials, then* Game0A *and* Game0B *are ε-statistically indistinguishable for $\varepsilon = \tilde{O}(\sqrt{n/m})$.*

Before proving the lemma, we will state and prove a useful combinatorial lemma about conditioning a pseudorandom function on a single input-output pair. Consider the following two ways of sampling a pseudorandom function.

Choose a pseudorandom function $\mathsf{PRF} : [m] \to [n]$
Let $s = |\mathsf{PRF}^{-1}(y)|$.
If $s = 0$, choose x at random from $[m]$, else choose a random x from $\mathsf{PRF}^{-1}(y)$.
Output (x, PRF).

Figure 18: $\mathsf{ExpA}[y]$

Choose a random $x \in [m]$
Choose a pseudorandom function $\mathsf{PRF} : [m] \to [n]$ so that $\mathsf{PRF}(x) = y$.
Let $s = |\mathsf{PRF}^{-1}(y)|$.
Output (x, PRF)

Figure 19: $\mathsf{ExpB}[y]$

Lemma A.7. *If* PRF *is pseudorandom, and m, n are polynomials, then for every $y \in [n]$,* $\mathsf{ExpA}[y]$ *and* $\mathsf{ExpB}[y]$ *are $\tilde{O}(\sqrt{n/m}) + \mathrm{negl}(\lambda)$ computationally indistinguishable.*

Proof of Lemma A.7. First, we will replace $\mathsf{PRF} : [m] \to [n]$ with a truly random function $f : [m] \to [n]$. We will argue later why using a pseudorandom function cannot increase the statistical distance between the two distributions by more than $\mathrm{negl}(\lambda)$.

Now, observe that in both experiments, the marginal distribution on x is uniform on $[m]$. Also, observe that for every fixed choice of x and s, the conditional distribution $f|x,s$ in each experiment is the same. Finally, note that the distribution on s is independent of x in each experiment. Thus, in order to bound the statistical distance between the two experiments, it suffices to bound the statistical distance between the marginal distributions of s in the two experiments.

In ExpA, the probability that $|f^{-1}(y)| = s$ is precisely the probability that y has exactly s preimages in a random function $f : [m] \to [n]$

$$\mathop{\mathbb{P}}_{f:[m]\to[n]}\left[|f^{-1}(y)| = s\right] = \binom{m}{s}\left(\frac{1}{n}\right)^s\left(1-\frac{1}{n}\right)^{m-s}$$

In ExpB, the probability that $|f^{-1}(y)| = s$ is precisely the probability that y has exactly $s-1$ preimages in a random function $f : [m-1] \to [n]$, since we fix the fact that $f(x) = y$ and the remainder of the function is chosen randomly.

$$\mathop{\mathbb{P}}_{f:[m-1]\to[n]}\left[|f^{-1}(y)| = s-1\right] = \binom{m-1}{s-1}\left(\frac{1}{n}\right)^{s-1}\left(1-\frac{1}{n}\right)^{m-s}$$

Thus, the statistical distance between the two distributions is

$$\begin{aligned}
&\sum_{s=1}^{m}\left|\binom{m}{s}\left(\frac{1}{n}\right)^{s}\left(1-\frac{1}{n}\right)^{m-s}-\binom{m-1}{s-1}\left(\frac{1}{n}\right)^{s-1}\left(1-\frac{1}{n}\right)^{m-s}\right| \\
&=\sum_{s=1}^{m}\left(\frac{1}{n}\right)^{s-1}\left(1-\frac{1}{n}\right)^{m-s}\left|\binom{m}{s}\left(\frac{1}{n}\right)-\binom{m-1}{s-1}\right| \\
&=\sum_{s=1}^{m}\binom{m-1}{s-1}\left(\frac{1}{n}\right)^{s-1}\left(1-\frac{1}{n}\right)^{m-s}\left|\frac{m}{sn}-1\right|
\end{aligned}$$

To tackle the final sum, we consider two cases roughly corresponding to whether $m/sn-1$ is close to 0 or far from 0. Typically, when we choose a random function from either distribution we will have a preimage size of $s \approx m/n$, in which case $|m/sn-1| \approx 0$. Since, by the binomial theorem,

$$\sum_{s=1}^{m}\binom{m-1}{s-1}(1/n)^{s-1}(1-1/n)^{m-s}=\sum_{s=0}^{m-1}\binom{m-1}{s}(1/n)^{s}(1-1/n)^{m-1-s}=1,$$

we will have that this portion of the sum is close to 0. In the atypical case, we will have that s is far from m/n, in this case we will use the fact that the probability of choosing a random function f with preimage size far from m/n is much smaller than n/m, and $|m/sn-1| \le m/n$, to conclude that this portion of the sum is also close to 0.

Specifically, fix some threshold τ and break the sum into two regions based on whether or not $s \in (1\pm\tau)m/n$.

$$\begin{aligned}
&\sum_{s=1}^{m}\binom{m-1}{s-1}\left(\frac{1}{n}\right)^{s-1}\left(1-\frac{1}{n}\right)^{m-s}\left|\frac{m}{sn}-1\right| \\
&=\sum_{s\in(1\pm\tau)m/n}\binom{m-1}{s-1}\left(\frac{1}{n}\right)^{s-1}\left(1-\frac{1}{n}\right)^{m-s}\left|\frac{m}{sn}-1\right|+\sum_{s\notin(1\pm\tau)m/n}\binom{m-1}{s-1}\left(\frac{1}{n}\right)^{s-1}\left(1-\frac{1}{n}\right)^{m-s}\left|\frac{m}{sn}-1\right| \\
&\le 2\tau\cdot\sum_{s\in(1\pm\tau)m/n}\binom{m-1}{s-1}\left(\frac{1}{n}\right)^{s-1}\left(1-\frac{1}{n}\right)^{m-s}+\frac{m}{n}\cdot\sum_{s\notin(1\pm\tau)m/n}\binom{m-1}{s-1}\left(\frac{1}{n}\right)^{s-1}\left(1-\frac{1}{n}\right)^{m-s} \\
&\le 2\tau+\frac{m}{n}\cdot\mathop{\mathbb{P}}_{f:[m-1]\to[n]}\left[|f^{-1}(y)|\notin(1\pm\tau)m/n\right] && \text{(Definition of Binomial Distribution)} \\
&\le 2\tau+\frac{m}{n}\cdot e^{-\Omega(\tau^2 m/n)}. && \text{(Chernoff bound)}
\end{aligned}$$

In this calculation, we make use of a form of the Chernoff bound that states if $X_1,\ldots,X_T$ are independent random variables taking values in $\{0,1\}$, and $X=\sum_{t=1}^{T}X_t$, then for every $\tau <= 1$, $\mathbb{P}\left[X\notin(1\pm\tau)\mathbb{E}[X]\right]\le e^{-\Omega(\tau^2\mathbb{E}[X])}$.

From this calculation, it is clear that there is a setting of $\tau=\tilde{O}(\sqrt{n/m})$ such that the final expression is bounded by $O(\tau)=\tilde{O}(\sqrt{n/m})$. Putting it together, the statistical distance between the two distributions in question is $\tilde{O}(\sqrt{n/m})+e^{-\Omega(m/n)}=\tilde{O}(\sqrt{n/m})$. This completes the proof.

Finally, we have to argue that the two distributions remain close if we use a pseudorandom function in place of a truly random function. Since m,n are polynomial, an efficient adversary can enumerate all of the input-output pairs of the function PRF. Thus, in order for PRF to be pseudorandom, the truth table of a random PRF must be computationally indistinguishable

from the truth table of a random function. By the above analysis, the distribution of the truth table of f depends only on x and s. Thus, for every fixed value of x, s, it must be that the distribution of $\mathsf{PRF} \mid x, s$ and the distribution of $f \mid x, s$ are $\varepsilon_{x,s}$ computationally indistinguishable for $\varepsilon_{x,s} = \mathrm{negl}(\lambda)/\mathbb{P}[x,s]$. Moreover, given (x, PRF), s is efficiently computable because m, n are polynomial. Thus, the two distributions are ε-computationally indistinguishable for some $\varepsilon \leq \sum_{x,s} \mathbb{P}[x,s](\mathrm{negl}(\lambda)/\mathbb{P}[x,s]) = mn \cdot \mathrm{negl}(\lambda) = \mathrm{negl}(\lambda)$.

Putting it together, we have that ExpA and ExpB are ε-computationally indistinguishable for $\varepsilon = \tilde{O}(\sqrt{n/m}) + \mathrm{negl}(\lambda)$, as desired. □

Now we return to proving Lemma A.6

Proof of Lemma A.6. First, fix any choice of (y_0, y_1, b_0, b_1). We want to show that the distributions on (x_0, x_1, PRF) in Game0A and Game0B are close. First consider just the distribution on $(x_0, \mathsf{PRF}) \mid (y_0, y_1, b_0, b_1)$. In Game0A, this distribution is exactly $\mathsf{ExpA}[y_{b_0}]$. In Game0B, this distribution is exactly $\mathsf{ExpB}[y_{b_0}]$. Thus, by Lemma A.7, the two distributions are ε-computationally indistinguishable for $\varepsilon = \tilde{O}(\sqrt{n/m}) + \mathrm{negl}(\lambda)$.

Now, we consider the case where $b_1 = b_0$. In this case, in both Game0A and Game0B, x_1 is a random preimage of $y_{b_1} = y_{b_0}$. Unless $x_1 = x_0$, x_1 is a uniformly random value in $[m]$. Since the collision probability is determined only by the number of preimages of y_{b_0}, Lemma A.7 also shows that the distribution on $x_1 \mid (y_0, y_1, b_0, b_1, x_0, \mathsf{PRF})$ is ε-computationally indistinguishable for $\varepsilon = \tilde{O}(\sqrt{n/m} + \mathrm{negl}(\lambda))$.

Now, we consider the case where $b_1 \neq b_0$. In this case the two values never collide, and once again the distribution of $x_1 \mid (y_0, y_1, b_0, b_1, x_0, \mathsf{PRF})$ is determined only by the number of preimages of y_{b_1}. Thus, we can again apply Lemma A.7 to argue that these two distributions are ε-computationally indistinguishable. This completes the proof of the Lemma. □

We can now state and prove the final step of the hybrid argument.

Lemma A.8. *If* PRF *is pseudorandom and m, n are polynomials, then* Game0B *and* Game1 *are ε-computationally indistinguishable for $\varepsilon = \tilde{O}(\sqrt{n/m})$.*

Proof. In either game, x_1 is uniformly random in $[m]$ if we condition on the event that $x_0 \neq x_1$. First consider the case where $b_0 = b_1$, then in Game0B then the probability of a collision is determined by the number of preimages of $y_{b_1} = y_{b_0}$, and by Lemma A.7 the collision probability is at most $\varepsilon = \tilde{O}(\sqrt{n/m}) + \mathrm{negl}(\lambda)$. However, in Game1 the probability of collision is exactly $1/m$. Now, in the case where $b_0 \neq b_1$, in Game0B the probability of collision is 0, whereas in Game1 the probability of collision is exactly $1/m$. Putting it together completes the proof of the Lemma. □

Combining the hybrids in Lemmata A.5, A.6,and A.8 completes the proof of Claim A.2. We remark that since m and n are polynomials in λ, the $\mathrm{negl}(\lambda)$ term is of a lower order than $\tilde{O}(\sqrt{n/m})$ so we are justified in dropping it from the asymptotic expression for the distinguishing probability.