On Differential Privacy and Adaptive Data Analysis with Bounded Space

Itai Dinur[1(✉)], Uri Stemmer[2,3], David P. Woodruff[4], and Samson Zhou[5,6]

[1] Ben-Gurion University, Be'er Sheva, Israel
dinuri@bgu.ac.il
[2] Tel Aviv University, Tel Aviv-Yafo, Israel
u@uri.co.il
[3] Google Research, Tel Aviv-Yafo, Israel
[4] Carnegie Mellon University, Pittsburgh, USA
dwoodruf@andrew.cmu.edu
[5] UC Berkeley, Berkeley, USA
[6] Rice University, Houston, USA

Abstract. We study the space complexity of the two related fields of *differential privacy* and *adaptive data analysis*. Specifically,

1. Under standard cryptographic assumptions, we show that there exists a problem P that requires exponentially more space to be solved efficiently with differential privacy, compared to the space needed without privacy. To the best of our knowledge, this is the first separation between the space complexity of private and non-private algorithms.
2. The line of work on adaptive data analysis focuses on understanding the number of *samples* needed for answering a sequence of adaptive queries. We revisit previous lower bounds at a foundational level, and show that they are a consequence of a space bottleneck rather than a sampling bottleneck.

To obtain our results, we define and construct an encryption scheme with multiple keys that is built to withstand a limited amount of key leakage in a very particular way.

1 Introduction

Query-to-communication lifting theorems allow translating lower bounds on the *query complexity* of a given function f to lower bounds on the *communication complexity* of a related function $\hat{f}$. Starting from the seminal work of Raz and McKenzie [31], several such lifting theorems were presented, and applied, to obtain new communication complexity lower bounds in various settings.

In the domain of cryptography, related results have been obtained, where the starting point is a lower bound on the *query complexity* of an adversary solving a cryptanalytic problem in an idealized model, such as the random oracle model [4].

The full version of this paper is available at https://eprint.iacr.org/2023/171.

C. Hazay and M. Stam (Eds.): EUROCRYPT 2023, LNCS 14006, pp. 35–65, 2023.
https://doi.org/10.1007/978-3-031-30620-4_2

The query complexity lower bound is then lifted to a *query-space* lower bound for a non-uniform (preprocessing) adversary solving the same problem [11,12,37].

Building on ideas developed in these lines of work, we present a new technique for translating sampling lower bounds to space lower bounds for problems in the context of *differential privacy* and *adaptive data analysis*. Before presenting our results, we motivate our settings.

1.1 Differential Privacy

Differential privacy [16] is a mathematical definition for privacy that aims to enable statistical analyses of datasets while providing strong guarantees that individual-level information does not leak. Informally, an algorithm that analyzes data satisfies differential privacy if it is robust in the sense that its outcome distribution does not depend "too much" on any single data point. Formally,

Definition 1.1 ([16]). *Let $\mathcal{A} : X^* \to Y$ be a randomized algorithm whose input is a dataset $D \in X^*$. Algorithm $\mathcal{A}$ is* (ε, δ)-differentially private (DP) *if for any two datasets D, D' that differ on one point (such datasets are called* neighboring*) and for any outcome set $F \subseteq Y$ it holds that* $\Pr[\mathcal{A}(D) \in F] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(D') \in F] + \delta$.

To interpret the definition, let D be a dataset containing n data points, each of which represents the information of one individual. Suppose that Alice knows all but one of these data points (say Bob's data point). Now suppose that we compute $z \leftarrow \mathcal{A}(D)$, and give z to Alice. If $\mathcal{A}$ is differentially private, then Alice learns very little about Bob's data point, because z would have been distributed roughly the same no matter what Bob's data point is.

Over the last few years, we have witnessed an explosion of research on differential privacy in various settings. In particular, a fruitful line of work has focused on designing differentially private algorithms with small *space* complexity, mainly in streaming settings. These works show many *positive* results and present differentially private algorithms with small space complexity for various problems. In fact, some of these works show that classical streaming algorithms are differentially private essentially *as is*. For example, Blocki et al. [5] show that the Johnson-Lindenstrauss transform itself preserves differential privacy, and Smith et al. [32] show this for the classical Flajolet-Martin Sketch.

In light of these positive results, one might think that algorithms with small space are particularly suitable for differential privacy, because these algorithms are not keeping too much information about the input to begin with.

Question 1.2. *Does differential privacy require more space?*

Our Results for Differential Privacy With bounded Space. We answer Question 1.2 in the affirmative, i.e., we show that differential privacy *may require more space*. To this end, we come up with a problem that can be solved using a small amount of space without privacy, but requires a large amount of space to be solved with privacy. As a first step, let us examine the following toy problem, which provides *some* answer to the above question.

A Toy Problem. Recall that F_2 (the second frequency moment of a stream) estimation with multiplicative approximation error $1+\alpha$ has an $\Omega(1/\alpha^2)$ space lower bound [38]. This immediately shows a separation for the problem of "output either the last element of the stream or a $(1+\alpha)$-approximation to the F_2 value of the stream". In the non-private setting, the last element can be output using space independent of α, but in the private setting the algorithm is forced to (privately) estimate F_2 and thus use at least $1/\alpha^2$ space. Of course, we could replace F_2 with other tasks that have a large space lower bound in the standard non-private model.

We deem this toy problem non-interesting because, at a high level, our goal is to show that there are cases where computing something privately requires a lot more space than computing "the same thing" non-privately. In the toy problem, however, the private and non-private algorithms are arguably *not* computing "the same thing". To reconcile this issue, we will focus on problems that are defined by a *function* (ranging over some metric space), and the desired task would be to approximate the value of this function. Note that this formulation disqualifies the toy problem from being a valid answer to Question 1.2, and that with this formulation there is a formal sense in which every algorithm for solving the task must compute (or approximate) "the same thing".

Let us make our setting more precise. In order to simplify the presentation, instead of studying the streaming model, we focus on the following computation model.[1] Consider an algorithm that is instantiated on a dataset D and then aims to answer a query with respect to D. We say that such an algorithm has space s if, before it gets the query, it shrinks D to a *summary* z containing at most s bits. Then, when obtaining the query, the algorithm answers it using only the summary z (without additional access to the original dataset D). Formally, we consider problems that are defined by a function $P : X^* \times Q \to M$, where X is the data domain, Q is a family of possible queries, and M is a metric space.

Definition 1.3. *We say that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ solves a problem $P : X^* \times Q \to M$ with space complexity s, sample complexity n, error α, and confidence β if*

1. *$\mathcal{A}_1 : X^* \to \{0,1\}^s$ is a* preprocessing procedure *that takes a dataset D and outputs an s-bit string.*
2. *For every input dataset $D \in X^n$ and every query $q \in Q$ it holds that*

$$\Pr_{\substack{z \leftarrow \mathcal{A}_1(D) \\ a \leftarrow \mathcal{A}_2(z,q)}} [|a - P(D,q)| \leq \alpha] \geq 1 - \beta.$$

We show the following theorem.

Theorem 1.4 (informal). *Let $d \in \mathbb{N}$ be a parameter controlling the size of the problem (specifically, data points from X can be represented using* $\mathrm{polylog}(d)$ *bits, and queries from Q can be represented using* $\mathrm{poly}(d)$ *bits). There exists a problem $P : X^* \times Q \to M$ such that the following holds.*

[1] We remark, however, that all of our results extend to the streaming setting. See Remark 3.3.

1. *P can be solved non-privately using* polylog(d) *bits of space.*
   % See Lemma 3.2 for the formal statement.
2. *P can be solved privately using sample and space complexity* $\tilde{O}(\sqrt{d})$.
   % See Lemma 3.4 for the formal statement.
3. *Assuming the existence of a sub-exponentially secure symmetric-key encryption scheme, every computationally-efficient differentially-private algorithm $\mathcal{A}$ for solving P must have space complexity $\tilde{\Omega}(\sqrt{d})$, even if its sample complexity is a large polynomial in d. Furthermore, this holds even if $\mathcal{A}$ is only required to be* computationally *differentially private (namely, the adversary we build against $\mathcal{A}$ is computationally efficient).*
   % See Corollary 3.13 for the formal statement.

Note that this is an exponential separation (in d) between the non-private space complexity and the private space complexity. We emphasize that the hardness of privately solving P does not come from not having enough samples. Indeed, by Item 2, $\tilde{O}(\sqrt{d})$ samples suffice for privately solving this problem. However, Item 3 states that unless the algorithm has large space, then it cannot privately solve this problem *even if it has many more samples than needed*.

To the best of our knowledge, this is the first result that separates the space complexity of private and non-private algorithms. Admittedly, the problem P we define to prove the above theorem is somewhat unnatural. In contrast, our negative results for adaptive data analysis (to be surveyed next) are for the canonical problem studied in the literature.

1.2 Adaptive Data Analysis

Consider a data analyst interested in testing a specific research hypothesis. The analyst acquires relevant data, evaluates the hypothesis, and (say) learns that it is false. Based on the findings, the analyst now decides on a second hypothesis to be tested, and evaluates it *on the same data* (acquiring fresh data might be too expensive or even impossible). That is, the analyst chooses the hypotheses *adaptively*, where this choice depends on previous interactions with the data. As a result, the findings are no longer supported by classical statistical theory, which assumes that the tested hypotheses are fixed before the data is gathered, and the analyst runs the risk of overfitting to the data.

Starting with [15], the line of work on *adaptive data analysis (ADA)* aims to design methods for provably guaranteeing statistical validity in such settings. Specifically, the goal is to design a mechanism $\mathcal{A}$ that initially obtains a dataset D containing t i.i.d. samples from some unknown distribution $\mathcal{P}$, and then answers k *adaptively chosen queries* w.r.t. $\mathcal{P}$. Importantly, $\mathcal{A}$'s answers must be accurate w.r.t. the underlying distribution $\mathcal{P}$, and not just w.r.t. the empirical dataset D. The main question here is,

Question 1.5. *How many samples does $\mathcal{A}$ need (i.e., what should t be) in order to support k such adaptive queries?*

As a way of dealing with worst-case analysts, the analyst is assumed to be adversarial in that it tries to cause the mechanism to fail. If a mechanism can maintain utility against such an adversarial analyst, then it maintains utility against any analyst. Formally, the canonical problem pursued by the line of work on ADA is defined as a two-player game between a mechanism $\mathcal{A}$ and an adversary $\mathcal{B}$. See Fig. 1.

1. The adversary $\mathcal{B}$ chooses a distribution $\mathcal{P}$ over a data domain X.
2. The mechanism $\mathcal{A}$ obtains a sample $S \sim \mathcal{P}^t$ containing t i.i.d. samples from $\mathcal{P}$.
3. For k rounds $j = 1, 2, \ldots, k$:
   - The adversary chooses a function $h_j : X \to \{-1, 0, 1\}$, possibly as a function of all previous answers given by the mechanism.
   - The mechanism obtains h_j and responds with an answer z_j, which is given to $\mathcal{B}$.

Fig. 1. A two-player game between a mechanism $\mathcal{A}$ and an adversary $\mathcal{B}$.

Definition 1.6 ([15]). *A mechanism $\mathcal{A}$ is (α, β)-accurate for k queries over a domain X using sample size t if for every adversary $\mathcal{B}$ (interacting with $\mathcal{A}$ in the game specified in Fig. 1) it holds that*

$$\Pr\left[\exists j \in [k] \text{ s.t. } |z_j - h_j(\mathcal{P})| > \alpha\right] \leq \beta,$$

where $h_j(\mathcal{P}) = \mathbb{E}_{x \sim \mathcal{P}}[h_j(x)]$ is the "true" value of h_j on the underlying distribution $\mathcal{P}$.

Following Dwork et al. [15], this problem has attracted a significant amount of work, most of which is focused on understanding how many *samples* are needed for adaptive data analysis (i.e., focused on Question 1.5). In particular, the following almost matching bounds are known.

Theorem 1.7 ([2,15]). *There exists a computationally efficient mechanism that is $(0.1, 0.1)$-accurate for k queries using sample size $t = \tilde{O}\left(\sqrt{k}\right)$.*

Theorem 1.8 ([24,34], informal). *Assuming the existence of one-way functions, every computationally efficient mechanism that is $(0.1, 0.1)$-accurate for k queries must have sample size at least $t = \Omega\left(\sqrt{k}\right)$.*

Our Results for Adaptive Data Analysis with Bounded Space. All prior work on the ADA problem treated it as a *sampling* problem, conveying the message that "adaptive data analysis requires more samples than non adaptive

data analysis". In this work we revisit the ADA problem at a foundational level, and ask:

Question 1.9. *Is there a more fundamental bottleneck for the ADA problem than the number of samples?*

Consider a mechanism $\mathcal{A}$ that initially gets the *full description of the underlying distribution* $\mathcal{P}$, but is required to shrink this description into a summary z, whose description length is identical to the description length of t samples from $\mathcal{P}$. (We identify the *space complexity* of $\mathcal{A}$ with the size of z in bits.) Afterwards, $\mathcal{A}$ needs to answer k adaptive queries using z, without additional access to $\mathcal{P}$. Does this give $\mathcal{A}$ more power over a mechanism that only obtains t samples from $\mathcal{P}$?

We show that, in general, the answer is no. Specifically, we show the following theorem.

Theorem 1.10 (informal version of Theorem 4.1). *Assuming the existence of one-way functions, then every computationally efficient mechanism that is* $(0.1, 0.1)$*-accurate for* k *queries must have* space complexity *at least* $\Omega\left(\sqrt{k}\right)$.

In fact, in the formal version of this theorem (see Theorem 4.1) we show that the space complexity must be at least $\Omega\left(\sqrt{k}\right)$ times the representation length of domain elements. We view this as a significant strengthening of the previous lower bounds for the ADA problem: it is not that the mechanism did not get enough information about $\mathcal{P}$; it is just that it cannot shrink this information in a way that allows for efficiently answering k adaptive queries. This generalizes the negative results of [24,34], as sampling $t = \sqrt{k}$ points from $\mathcal{P}$ is just one particular way for storing information about $\mathcal{P}$.

1.3 Our Techniques

We obtain our results through a combination of techniques across several research areas including cryptography, privacy, learning theory, communication complexity, and information theory.

Our Techniques: Multi-instance Leakage-Resilient Scheme. The main cryptographic tool we define and construct is a suitable encryption scheme with multiple keys that is built to withstand a certain amount of key leakage in a very particular way. Specifically, the scheme consists of n instances of an underlying encryption scheme with independent keys (each of length λ bits). The keys are initially given to an adversary who shrinks them down to a summary z containing $s \ll n \cdot \lambda$ bits. After this phase, each instance independently sets an additional parameter, which is public but unknown to the adversary in the initial phase. Then, the adversary obtains encryptions of plaintexts under the n keys. We require that given z and the public parameters, the plaintexts encrypted with

each key remain computationally hidden, except for a small number of the keys (which depends on s, but not on n).

We call the scheme a *multi-instance leakage-Resilient scheme* (or MILR scheme) to emphasize the fact that although the leakage of the adversary is an arbitrary function of all the n keys, the scheme itself is composed of n instances that are completely independent.

The efficiency of the MILR scheme is measured by two parameters: (1) the number of keys under which encryptions are (potentially) insecure, and (2) the loss in the security parameter λ. The scheme we construct is optimal in both parameters up to a multiplicative constant factor. First, encryptions remain hidden for all but $O\left(\frac{s}{\lambda}\right)$ of the keys. This is essentially optimal, as the adversary can define z to store the first $\frac{s}{\lambda}$ keys. Second, we lose a constant factor in the security parameter λ. An additional advantage of our construction is that its internal parameters do not depend on s. If we did allow such a dependency, then in some settings (particularly when $s \leq o(n \cdot \lambda)$) it would be possible to fine-tune the scheme to obtain a multiplicative $1 + o(1)$ loss in the efficiency parameters, but this has little impact on our application.

Our construction is arguably the most natural one. To encrypt a plaintext with a λ-bit key after the initial phase, we first apply an extractor (with the public parameter as a seed) to hash it down to a smaller key, which is used to encrypt the plaintext with the underlying encryption scheme.

The MILR scheme is related to schemes developed in the area of leakage-resilient cryptography (cf., [10,19,20,26,27,30,33]) where the basic technique of randomness extraction is commonly used. However, leakage-resilient cryptography mainly deals with resilience of cryptosystems to side-channel attacks, whereas our model is not designed to formalize security against such attacks and has several properties that are uncommon in this domain (such as protecting independent multiple instances of an encryption scheme in a way that inherently makes some of them insecure). Consequently, the advanced cryptosytems developed in the area of leakage-resilient cryptography are either unsuitable, overly complex, or inefficient for our purposes.

Despite the simplicity of our construction, our proof that it achieves the claimed security property against leakage is somewhat technical. We stress that we do not rely on hardness assumptions for specific problems, nor assume that the underlying encryption scheme has special properties such as resilience to related-key attacks. Instead, our proof is based on the pre-sampling technique introduced by Unruh [37] to prove security of cryptosystems against non-uniform adversaries in the random oracle model. This technique has been recently refined and optimized in [11,12,14] based on tools developed in the area of communication complexity [22,28]. The fact that we use the technique to prove security in a computational (rather than information-theoretic) setting seems to require the assumption that the underlying encryption scheme is secure against non-uniform adversaries (albeit this is considered a standard assumption).

Our Techniques: Privacy Requires More Space. We design a problem that can be solved non-privately with very small space complexity, but requires a large space complexity with privacy. To achieve this, we lift a known negative result on the *sample* complexity of privately solving a specific problem to obtain a *space* lower bound for a related problem. The problem we start with, for which there exists a sampling lower bound, is the so-called *1-way marginals* problem with parameter d. In this problem, our input is a dataset $D \in (\{0,1\}^d)^*$ containing a collection of binary vectors, each of length d. Our goal is to output a vector $\vec{a} \in [0,1]^d$ that approximates the *average* of the input vectors to within small L_∞ error, say to within error $1/10$. That is, we want vector $\vec{a} \in [0,1]^d$ to satisfy: $\left\| \vec{a} - \frac{1}{|D|} \sum_{\vec{x} \in D} \vec{x} \right\|_\infty \leq \frac{1}{10}$.

We say that an algorithm for this problem has *sample complexity* n if, for every input dataset of size n, it outputs a good solution with probability at least 0.9. One of the most fundamental results in the literature of differential privacy shows that this problem requires a large dataset:

Theorem 1.11 ([9], informal). *Every differentially private algorithm for solving the 1-way marginal problem with parameter d must have sample complexity $n = \Omega(\sqrt{d})$.*

To lift this sampling lower bound into a space lower bound, we consider a problem in which the input dataset contains n *keys* $\vec{x} = (x_1, \ldots, x_n)$ (sampled from our MILR scheme). The algorithm must then shrink this dataset into a summary z containing s bits. Afterwards, the algorithm gets a "query" that is specified by a collection of n ciphertexts, each of which is an encryption of a d-bit vector. The desired task is to approximate the average of the *plaintext* input vectors. Intuitively, if the algorithm has space $s \ll \sqrt{d}$, then by the properties of our MILR scheme, it can decrypt at most $\approx s \ll \sqrt{d}$ of these d-bit vectors, and is hence trying to solve the 1-way marginal problem with fewer than $\sqrt{d}$ samples. We show that this argument can be formalized to obtain a contradiction.

Our Techniques: ADA Is About Space. As we mentioned, Hardt and Ullman [24] and Steinke and Ullman [34] showed that the ADA problem requires a large *sample* complexity (see Theorem 1.8). Specifically, they showed that there exists a computationally efficient adversary that causes every efficient mechanism to fail in answering adaptive queries. Recall that the ADA game (see Fig. 1) begins with the adversary choosing the underlying distribution.

We lift the negative result of [24,34] to a *space* lower bound. To achieve this, we design an alternative adversary that first samples a large collection of *keys* for our MILR scheme, and then defines the target distribution to be uniform on these sampled keys. Recall that in our setting, the mechanism gets an exact description of this target distribution and afterwards it must shrink this description into s bits. However, by the properties of our MILR scheme, this means that the mechanism would only be able to decrypt ciphertexts that correspond to at most $\approx s/\lambda$ of the keys. We then show that the adversary

of [24,34] can be simulated under these conditions, where the "input sample" from their setting corresponds to the collection of indices of keys for which the mechanism can decrypt ciphertexts.

1.4 Applications to Communication Complexity

Finally, our arguments also provide distributional one-way communication complexity lower bounds, which are useful when the goal is to compute a relation with a very low success probability. To the best of our knowledge, existing query-to-communication lifting theorems, e.g., [22,23,31] do not consider the problems and input distributions that we consider here. Roughly speaking, we show that if any sampling based protocol for computing a function f requires k samples $(a_1, b_1), \ldots, (a_k, b_k)$, where $a_i \in \{0,1\}^t$ for each $i \in [k]$, then any one-way protocol that computes $f(A, B)$ in this setting must use $\Omega(kt)$ communication.

More precisely, in the two-player one-way communication game, inputs A and B are given to Alice and Bob, respectively, and the goal is for Alice to send a minimal amount of information to Bob, so that Bob can compute $f(A, B)$ for some predetermined function f. The communication cost of a protocol Π is the size of the largest message in bits sent from Alice across all possible inputs A and the (randomized) communication complexity is the minimum communication cost of a protocol that succeeds with probability at least $\frac{2}{3}$. In the distributional setting, A and B are further drawn from a known underlying distribution.

In our distributional setting, suppose Alice has m independent and uniform numbers $a_1, \ldots, a_m$ so that either $a_i \in GF(p)$ for all $i \in [m]$ or $a_i \in GF(2^t)$ for sufficiently large t for all $i \in [m]$ and suppose Bob has m independent and uniform numbers $b_1, \ldots, b_m$ from the same field, either $GF(p)$ or $GF(2^t)$. Then for any function $f(\langle a_1, b_1\rangle, \ldots, \langle a_m, b_m\rangle)$, where the dot products are taken over $GF(2)$ or $f(a_1 \cdot b_1, \ldots, a_m \cdot b_m)$, where the products are taken over $GF(p)$, has the property that the randomized one-way communication complexity of computing f with probability σ is the same as the number of samples from $a_1, \ldots, a_m$ that Alice needs to send Bob to compute f with probability $\sigma - \varepsilon$. It is easy to prove sampling lower bounds for many of these problems, sum as $\sum_i a_i \cdot b_i \pmod p$ or $\mathrm{MAJ}(\langle a_1, b_1\rangle, \ldots, \langle a_m, b_m\rangle)$, and this immediately translates into communication complexity lower bounds. The main intuition for our overall lower bound argument is that the numbers $b_1, \ldots, b_m$ can be viewed as the hash functions that Bob has and thus we can apply a variant of the leftover hash lemma if Bob only has a small subset of these numbers. See the full version of this work for the details.

1.5 Related Works

Dwork et al. [17] used *traitor-tracing schemes* to prove computational sampling lower bounds for differential privacy. Their results were extended by Ullman [36], who used *fingerprinting codes* to construct a novel traitor-tracing scheme and to obtain stronger computational sampling lower bounds for differential privacy. Ullman's construction can be viewed as an encrypted variant of the 1-way

marginal problem. Bun et al. [8] and Alon et al. [1] showed that there are trivial learning tasks that require asymptotically more samples to solve with differential privacy (compared to the non-private sample complexity). These results are fundamentally different than ours, as they are about sampling rather than space. Feldman [21] and Brown et al. [7] showed that there are learning problems for which every *near optimal* learning algorithm (obtaining near optimal error w.r.t. to the number of input *samples* it takes) must memorize a large portion of its input data. These works do not directly address the *additional* space required for preserving privacy. See the full version for additional related works.

1.6 Paper Structure

The rest of the paper is structured as follows. The MILR scheme is defined in Sect. 2. Our results for differential privacy and adaptive data analysis are described in Sects. 3 and 4, respectively. We construct our MILR scheme in Sect. 5, and prove its security in Sect. 6. Some of the technical details from these sections are deferred to the full version of this work.

2 Multi-instance Leakage-Resilient Scheme

We define a *multi-instance leakage-resilient scheme* (or MILR scheme) to be a tuple of efficient algorithms (Gen, Param, Enc, Dec) with the following syntax:

- Gen is a randomized algorithm that takes as input a security parameter λ and outputs a λ-bit secret key. Formally, $x \leftarrow \text{Gen}(1^\lambda)$.
- Param is a randomized algorithm that takes as input a security parameter λ and outputs a $\text{poly}(\lambda)$-bit public parameter. Formally, $p \leftarrow \text{Param}(1^\lambda)$.
- Enc is a randomized algorithm that takes as input a secret key x, a public parameter p, and a message $m \in \{0,1\}$, and outputs a ciphertext $c \in \{0,1\}^{\text{poly}(\lambda)}$. Formally, $c \leftarrow \text{Enc}(x,p,m)$.
- Dec is a deterministic algorithm that takes as input a secret key x, a public parameter p, and a ciphertext c, and outputs a decrypted message m'. If the ciphertext c was an encryption of m under the key x with the parameter p, then $m' = m$. Formally, if $c \leftarrow \text{Enc}(x,p,m)$, then $\text{Dec}(x,p,c) = m$ with probability 1.

To define the security of an MILR scheme, let $n \in \mathbb{N}$, let $\vec{x} = (x_1, \ldots, x_n)$ be a vector of keys, and let $\vec{p} = (p_1, \ldots, p_n)$ be a vector of public parameters (set once for each scheme by invoking Param). Let $J \subseteq [n]$ be a subset, referred to as the "hidden coordinates". Now consider a pair of oracles $\mathcal{E}_1(\vec{x}, \vec{p}, J, \cdot, \cdot)$ and $\mathcal{E}_0(\vec{x}, \vec{p}, J, \cdot, \cdot)$ with the following properties.

1. $\mathcal{E}_1(\vec{x}, \vec{p}, J, \cdot, \cdot)$ takes as input an index of a key $j \in [n]$ and a message m, and returns $\text{Enc}(x_j, p_j, m)$.
2. $\mathcal{E}_0(\vec{x}, \vec{p}, J, \cdot, \cdot)$ takes the same inputs. If $j \in J$ then the outcome is $\text{Enc}(x_j, p_j, 0)$, and otherwise the outcome is $\text{Enc}(x_j, p_j, m)$.

Definition 2.1. *Let λ be a security parameter. Let $\Gamma : \mathbb{R} \to \mathbb{R}$ and $\overline{\tau} : \mathbb{R}^2 \to \mathbb{R}$ be real-valued functions. An MILR scheme* $(\mathrm{Gen}, \mathrm{Param}, \mathrm{Enc}, \mathrm{Dec})$ *is* $(\Gamma, \overline{\tau})$-secure *against space bounded preprocessing adversaries if the following holds.*

(1) Multi semantic security: *For every $n = \mathrm{poly}(\Gamma(\lambda))$ and every $\mathrm{poly}(\Gamma(\lambda))$-time adversary $\mathcal{B}$ there exists a negligible function* negl *such that*

$$\left| \Pr_{\vec{x},\vec{p},\mathcal{B},\mathrm{Enc}} \left[\mathcal{B}^{\mathcal{E}_0(\vec{x},\vec{p},[n],\cdot,\cdot)}(\vec{p}) = 1 \right] - \Pr_{\vec{x},\vec{p},\mathcal{B},\mathrm{Enc}} \left[\mathcal{B}^{\mathcal{E}_1(\vec{x},\vec{p},[n],\cdot,\cdot)}(\vec{p}) = 1 \right] \right| \leq \mathrm{negl}(\Gamma(\lambda)).$$

That is, a computationally bounded adversary that gets the public parameters, but not the keys, cannot tell whether it is interacting with $\mathcal{E}_0$ or with $\mathcal{E}_1$.

(2) Multi-security against a bounded preprocessing adversary: *For every $n = \mathrm{poly}(\Gamma(\lambda))$, every $s \leq n \cdot \lambda$, and every preprocessing procedure $F : \left(\{0,1\}^\lambda\right)^n \to \{0,1\}^s$ (possibly randomized), there exists a random function $J = J(F, \vec{x}, z, \vec{p}) \subseteq [n]$ that given a collection of keys $\vec{x}$ and public parameters $\vec{p}$, and an element $z \leftarrow F(\vec{x})$, returns a subset of size $|J| \geq \tau := n - \overline{\tau}(\lambda, s)$ such that for every $\mathrm{poly}(\Gamma(\lambda))$-time algorithm $\mathcal{B}$ there exists a negligible function* negl *satisfying*

$$\left| \Pr_{\substack{\vec{x},\vec{p},\mathcal{B},\mathrm{Enc} \\ z \leftarrow F(\vec{x}) \\ J \leftarrow J(F,\vec{x},z,\vec{p})}} \left[\mathcal{B}^{\mathcal{E}_0(\vec{x},\vec{p},J,\cdot,\cdot)}(z,\vec{p}) = 1 \right] - \Pr_{\substack{\vec{x},\vec{p},\mathcal{B},\mathrm{Enc} \\ z \leftarrow F(\vec{x}) \\ J \leftarrow J(F,\vec{x},z,\vec{p})}} \left[\mathcal{B}^{\mathcal{E}_1(\vec{x},\vec{p},J,\cdot,\cdot)}(z,\vec{p}) = 1 \right] \right| \leq \mathrm{negl}(\Gamma(\lambda)).$$

That is, even if s bits of our n keys were leaked (computed by the preprocessing function F operating on the keys), then still encryptions w.r.t. the keys of J are computationally indistinguishable.

Remark 2.2. *When Γ is the identity function, we simply say that the scheme is $\overline{\tau}$-secure. Note that in this case, security holds against all adversaries with runtime polynomial in the security parameter λ. We will further assume the existence of a sub-exponentially secure encryption scheme. By that we mean that there exists a constant $\nu > 0$ such that the scheme is $(\Gamma, \overline{\tau})$-secure for $\Gamma(\lambda) = 2^{\lambda^\nu}$. That is, we assume the existence of a scheme in which security holds against all adversaries with runtime polynomial in 2^{λ^ν}.*

In Sects. 5 and 6 we show the following theorem.

Theorem 2.3. *Let $\Omega(\lambda) \leq \Gamma(\lambda) \leq 2^{o(\lambda)}$. If there exists a $\Gamma(\lambda)$-secure encryption scheme against non-uniform adversaries then there exists an MILR scheme that is* $(\Gamma(\lambda), \overline{\tau})$-secure *against space bounded non-uniform preprocessing adversaries, where $\overline{\tau}(\lambda, s) = \frac{2s}{\lambda} + 4$.*

3 Space Hardness for Differential Privacy

Consider an algorithm that is instantiated on a dataset D and then aims to answer a query w.r.t. D. We say that such an algorithm has space s if, before it gets the query, it shrinks D to a *summary* z containing at most s bits. Then, when obtaining the query, the algorithm answers it using only the summary z (without additional access to the original dataset D).

Let $(\mathrm{Gen}, \mathrm{Param}, \mathrm{Enc}, \mathrm{Dec})$ be an MILR scheme with security parameter λ. In the (λ, d)-decoded average (DA) problem with sample complexity n, the input dataset contains n keys, that is $D = (x_1, \ldots, x_n) \in \left(\{0,1\}^\lambda\right)^n$. A query $q = ((p_1, c_1), \ldots, (p_n, c_n))$ is specified using n pairs (p_i, c_i) where p_i is a public parameter and c_i is a ciphertext, which is an encryption of a binary vector of length d. The goal is to release a vector $\vec{a} = (a_1, \ldots, a_d) \in [0,1]^d$ that approximates the "decrypted average vector (dav)", defined as $\mathrm{dav}_q(D) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Dec}(x_i, p_i, c_i)$.

Definition 3.1. *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an algorithm where $\mathcal{A}_1 : \left(\{0,1\}^\lambda\right)^n \to \{0,1\}^s$ is the preprocessing procedure that summarizes a dataset of n keys into s bits, and where $\mathcal{A}_2$ is the "response algorithm" that gets the outcome of $\mathcal{A}_1(D)$ and a query q. We say that $\mathcal{A}$ solves the DA problem if with probability at least $9/10$ the output is a vector $\vec{a}$ satisfying $\|\vec{a} - \mathrm{dav}_q(D)\|_\infty \leq \frac{1}{10}$.*

Without privacy considerations, the DA problem is almost trivial. Specifically,

Lemma 3.2. *The (λ, d)-DA problem can be solved efficiently using space $s = O(\lambda \log(d))$.*

Proof. The preprocessing algorithm $\mathcal{A}_1$ samples $O(\log d)$ of the input keys. Algorithm $\mathcal{A}_2$ then gets the query q and estimates the dav vector using the sampled keys. The lemma then follows by the Chernoff bound.

Remark 3.3. *As we mentioned, to simplify the presentation, in our computational model we identify the space complexity of algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with the size of the output of algorithm $\mathcal{A}_1$. We remark, however, that our separation extends to a streaming model where both $\mathcal{A}_1$ and $\mathcal{A}_2$ are required to have small space. To see this, note that algorithm $\mathcal{A}_1$ in the above proof already has small space (and not just small output length), as it merely samples $O(\log d)$ keys from its input dataset. We now analyze the space complexity of $\mathcal{A}_2$, when it reads the query q in a streaming fashion. Recall that the query q contains n public parameters $p_1, \ldots, p_n$ and n ciphertexts $c_1, \ldots, c_n$, where each c_i is an encryption of a d-bit vector, call it $y_i \in \{0,1\}^d$. To allow $\mathcal{A}_2$ to read q using small space, we order it as follows: $q = (p_1, \ldots, p_n), (c_{1,1}, \ldots, c_{n,1}), \ldots, (c_{1,d}, \ldots, c_{n,d}) \triangleq q_0 \circ q_1 \circ \cdots \circ q_d$. Here $c_{i,j} = \mathrm{Enc}(x_i, p_i, y_i[j])$ is an encryption of the jth bit of y_i using key x_i and public parameter p_i. Note that the first part of the stream, q_0, contains the public parameters, and then every part q_j contains encryptions of the jth bit of each of the n input vectors. With this ordering of the query, algorithm $\mathcal{A}_2$ begins by reading q_0 and storing the $O(\log d)$ public parameters corresponding to the keys that were stored by $\mathcal{A}_1$. Then, for every $j \in [d]$, when reading q_j, algorithm $\mathcal{A}_2$*

estimates the average of the jth coordinate using the sampled keys. Algorithm $\mathcal{A}_2$ then outputs this estimated value, and proceeds to the next coordinate. This can be implemented using space complexity $\mathrm{poly}(\lambda \log(d))$.

So, without privacy constraints, the DA problem can be solved using small space. We now show that, assuming that the input dataset is large enough, the DA problem can easily be solved with differential privacy using *large* space. Specifically,

Lemma 3.4. *There is a computationally efficient (ε, δ)-differentially private algorithm that solves the (λ, d)-DA problem using space $s = O\left(\frac{1}{\varepsilon} \cdot \sqrt{d \cdot \log(\frac{1}{\delta})} \cdot \lambda \cdot \log d\right)$, provided that the size of the input dataset satisfies $n = \Omega(s)$ (large enough).*

Proof. The preprocessing algorithm $\mathcal{A}_1$ samples $\approx \sqrt{d}$ of the keys. By standard composition theorems for differential privacy [18], this suffices for the response algorithm $\mathcal{A}_2$ to privately approximate each of the d coordinates of the target vector.

Thus the DA problem can be solved non-privately using small space, and it can be solved privately using large space. Our next goal is to show that large space is indeed necessary to solve this problem privately. Before showing that, we introduce several additional preliminaries on computational differential privacy and on fingerprinting codes.

3.1 Preliminaries on Computational Differential Privacy and Fingerprinting Codes

Computational differential privacy was defined by Beimel et al [3] and Mironov et al. [29]. Let $\mathcal{A}$ be a randomized algorithm (mechanism) that operates on datasets. Computational differential privacy is defined via a two player game between a *challenger* and an adversary, running a pair of algorithms (Q, T). The game begins with the adversary Q choosing a pair of neighboring datasets (D_0, D_1) of size n each, as well as an arbitrary string r (which we think of as representing its internal state). Then the challenger samples a bit b and applies $\mathcal{A}(D_b)$ to obtain an outcome a. Then $T(r, \cdot)$ is applied on a and tries to guess b. Formally,

Definition 3.5. *Let λ be a security parameter, let ε be a constant, and let $\delta : \mathbb{R} \to \mathbb{R}$ be a function. A randomized algorithm $\mathcal{A} : X^* \to Y$ is* (ε, δ)-computationally differentially private (CDP) *if for every $n = \mathrm{poly}(\lambda)$ and every non-uniform* $\mathrm{poly}(\lambda)$*-time adversary (Q, T) there exists a negligible function* negl *such that*

$$\Pr_{\substack{(r,D_0,D_1)\leftarrow Q \\ \mathcal{A},T}}[T(r, \mathcal{A}(D_0)) = 1] \leq e^{\varepsilon} \cdot \Pr_{\substack{(r,D_0,D_1)\leftarrow Q \\ \mathcal{A},T}}[T(r, \mathcal{A}(D_1)) = 1] + \delta(n) + \mathrm{negl}(\lambda).$$

Definition 3.6. *Let ε be a constant and let $\delta = \delta(\lambda)$ be a function. Given two probability ensembles $\mathcal{X} = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ we write $\mathcal{X} \approx_{\varepsilon,\delta} \mathcal{Y}$ if for*

every non-uniform probabilistic polynomial-time distinguisher D there exists a negligible function negl *such that* $\Pr_{x \leftarrow X_\lambda}[D(x) = 1] \leq e^{\varepsilon} \cdot \Pr_{y \leftarrow Y_\lambda}[D(y) = 1] + \delta(\lambda) + \text{negl}(\lambda)$, *and vice versa.*

We recall the concept of *fingerprinting codes*, which was introduced by Boneh and Shaw [6] as a cryptographic tool for watermarking digital content. Starting from the work of Bun, Ullman, and Vadhan [9], fingerprinting codes have played a key role in proving lower bounds for differential privacy in various settings.

A (collusion-resistant) fingerprinting code is a scheme for distributing codewords $w_1, \cdots, w_n$ to n users that can be uniquely traced back to each user. Moreover, if a group of (at most k) users combines its codewords into a pirate codeword $\hat{w}$, then the pirate codeword can still be traced back to one of the users who contributed to it. Of course, without any assumption on how the pirates can produce their combined codeword, no secure tracing is possible. To this end, the pirates are constrained according to a *marking assumption*, which asserts that the combined codeword must agree with at least one of the "real" codewords in each position. Namely, at an index j where $w_i[j] = b$ for every $i \in [n]$, the pirates are constrained to output $\hat{w}$ with $\hat{w}[j] = b$ as well.[2]

Definition 3.7 ([6,35]). *A k-collusion resilient fingerprinting code of length d for n users with failure probability γ, or (n, d, k, γ)-FPC in short, is a pair of random variables $C \in \{0,1\}^{n \times d}$ and* Trace $: \{0,1\}^d \to 2^{[n]}$ *such that the following holds. For all adversaries $P : \{0,1\}^{k \times d} \to \{0,1\}^d$ and $S \subseteq [n]$ with $|S| = k$,*

$$\Pr_{C,\text{Trace},P}[(\forall 1 \leq j \leq d \ \ \exists i \in [n] \ \textit{s.t.}\ P(C_S)[j] = c_i[j]) \wedge (\text{Trace}(P(C_S)) = \emptyset)] \leq \gamma,$$

and

$$\Pr_{C,\text{Trace},P}[\text{Trace}(P(C_S)) \cap ([n] \setminus S) \neq \emptyset] \leq \gamma,$$

where C_S contains the rows of C given by S.

Remark 3.8. *As mentioned, the condition $\{\forall 1 \leq j \leq d \ \ \exists i \in [n]$ s.t. $P(C_S)[j] = c_i[j]\}$ is called the "marking assumption". The second condition is called the "small probability of false accusation". Hence, if the adversary P guarantees that its output satisfies the marking assumption, then with probability at least $1 - 2\gamma$ it holds that algorithm* Trace *outputs an index $i \in S$.*

Theorem 3.9 ([6,34,35]). *For every $1 \leq k \leq n$ there is a k-collusion-resilient fingerprinting code of length $d = O(k^2 \cdot \log n)$ for n users with failure probability $\gamma = \frac{1}{n^2}$ and an efficiently computable Trace function.*

We remark that there exist both adaptive and non-adaptive constructions of fingerprinting codes with the guarantees of Theorem 3.9; we use the non-adaptive variant.

[2] We follow the formulation of the marking assumption as given by [34], which is a bit different than the commonly considered one.

3.2 A Negative Result for the DA Problem

Our main negative result for space bounded differentially private algorithms is the following.

Theorem 3.10. *Let* $\Pi = (\mathrm{Gen}, \mathrm{Param}, \mathrm{Enc}, \mathrm{Dec})$ *be an MILR scheme with security parameter* λ *that is* $(\Gamma, \overline{\tau})$*-secure against space bounded preprocessing adversaries. Let* $d \leq \mathrm{poly}(\Gamma(\lambda))$ *and* $n \leq \mathrm{poly}(\Gamma(\lambda))$ *be functions of* λ*. Let* ε *be a constant and let* $\delta \leq \frac{1}{4n(e^{\varepsilon}+1)} = \Theta(\frac{1}{n})$*. For every* $\mathrm{poly}(\Gamma(\lambda))$*-time* (ε, δ)*-CDP algorithm for the* (λ, d)*-DA problem with sample complexity* n *and space complexity* s *it holds that* $\overline{\tau}(\lambda, s) = \Omega\left(\sqrt{\frac{d}{\log n}}\right)$.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a $\mathrm{poly}(\Gamma(\lambda))$-time CDP algorithm for the (λ, d)-DA problem using sample complexity $n = \mathrm{poly}(\Gamma(\lambda))$ and space complexity s. Denote $\overline{\tau} = \overline{\tau}(\lambda, s)$, and assume towards a contradiction that $\overline{\tau} = O\left(\sqrt{\frac{d}{\log n}}\right)$ (small enough). We construct the following adversary $\mathcal{B}$ to an $\left(n+1, d, \overline{\tau}, \frac{1}{n^2}\right)$-FPC (such a code is guaranteed to exist by Theorem 3.9 and by the contradictory assumption).

1. The input is n codewords $w_1, \ldots, w_n \in \{0,1\}^d$.
2. Sample n keys $x_1, \ldots, x_n \sim \mathrm{Gen}(1^{\lambda})$.
3. Let $z \leftarrow \mathcal{A}_1(x_1, \ldots, x_n)$.
4. Sample n public parameters $p_1, \ldots, p_n \sim \mathrm{Param}(1^{\lambda})$.
5. For $i \in [n]$ let $c_i \leftarrow \mathrm{Enc}(x_i, p_i, w_i)$.
6. Let $\vec{a} \leftarrow \mathcal{A}_2(z, (p_1, c_1), \ldots, (p_n, c_n))$.
7. Output $\vec{a}$, after rounding its coordinates to $\{0,1\}$.

We think of $\mathcal{B}$ as an adversary to an FPC, and indeed, its input is a collection of codewords and its output is a binary vector of length d. Observe that if $\mathcal{A}$ solves the DA problem (i.e., approximates the decrypted average vector), then for every coordinate, the outcome of $\mathcal{B}$ must agree with at least one of the input codewords, namely, it satisfies the marking assumption (see Remark 3.8).

Remark 3.11. *Before we proceed with the formal proof, we give an overview of its structure (this remark can be ignored, and is not needed for the formal proof). Informally, we will show that*

(1) Algorithm $\mathcal{B}$ *is computationally differentially private w.r.t. the collection of* codewords *(even though our assumption on* $\mathcal{A}$ *is that it is private w.r.t. the* keys*).*

(2) Leveraging the properties of the MILR scheme, we will show that $\mathcal{B}$ *must effectively ignore most of its inputs, except for at most* $\overline{\tau}$ *codewords. This means that* $\mathcal{B}$ *is effectively an FPC adversary that operates on only* $\overline{\tau}$ *codewords (rather than the* n *codewords it obtains as input).*

(3) A known result in the literature of differential privacy (starting from [9]) is that a successful FPC adversary cannot be differentially private, because this would contradict the fact that the tracing algorithm is able to recover one of its input points. Our gain here comes from the fact that $\mathcal{B}$ only uses (effectively) $\overline{\tau}$ codewords, and hence, in order to get a contradiction, it suffices to use an FPC with a much shorter codeword-length (only $\approx \overline{\tau}^2$ instead of $\approx n^2$). This will mean that the hardness of the DA problem depends on the space of $\mathcal{A}$ (which controls $\overline{\tau}$) rather than the size of the input (which is n).

Recall that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is computationally differentially private w.r.t. the *keys*. We first show that $\mathcal{B}$ is computationally differentially private w.r.t. the *codewords*. To this end, let Q be an adversary (as in Definition 3.5) that outputs a pair of neighboring datasets $(\vec{w}, \vec{w'})$, each containing n codewords, together with a state r. Given $(\vec{w}, \vec{w'})$, we write $\ell = \ell(\vec{w}, \vec{w'}) \subseteq [n]$ to denote the index on which $\vec{w}, \vec{w'}$ differ. We also write x_0 to denote another key, independent of the keys $x_1, \ldots, x_n$ sampled by algorithm $\mathcal{B}$. By the privacy guarantees of algorithm $\mathcal{A}$ and by the semantic security of the encryption scheme (see Definition 2.1) we have that

$$
\begin{aligned}
&\langle r, \mathcal{B}(\vec{w})\rangle \equiv \\
&\quad\equiv \langle r, \mathcal{A}_2\left(\mathcal{A}_1(x_1, ..., x_\ell, ..., x_n), \vec{p}, \mathrm{Enc}(x_1, p_1, w_1), ..., \mathrm{Enc}(x_\ell, p_\ell, w_\ell), ..., \mathrm{Enc}(x_n, p_n, w_n)\right)\rangle \\
&\approx_{(\varepsilon,\delta)} \langle r, \mathcal{A}_2\left(\mathcal{A}_1(x_1, ..., x_0, ..., x_n), \vec{p}, \mathrm{Enc}(x_1, p_1, w_1), ..., \mathrm{Enc}(x_\ell, p_\ell, w_\ell), ..., \mathrm{Enc}(x_n, p_n, w_n)\right)\rangle \\
&\quad\equiv_c \langle r, \mathcal{A}_2\left(\mathcal{A}_1(x_1, ..., x_0, ..., x_n), \vec{p}, \mathrm{Enc}(x_1, p_1, w_1), ..., \mathrm{Enc}(x_\ell, p_\ell, w'_\ell), ..., \mathrm{Enc}(x_n, p_n, w_n)\right)\rangle \\
&\approx_{(\varepsilon,\delta)} \langle r, \mathcal{A}_2\left(\mathcal{A}_1(x_1, ..., x_\ell, ..., x_n), \vec{p}, \mathrm{Enc}(x_1, p_1, w_1), ..., \mathrm{Enc}(x_\ell, p_\ell, w'_\ell), ..., \mathrm{Enc}(x_n, p_n, w_n)\right)\rangle \\
&\quad\equiv \langle r, \mathcal{B}(\vec{w'})\rangle.
\end{aligned}
$$

So algorithm $\mathcal{B}$ is $(2\varepsilon, (e^\varepsilon + 1)\delta)$-computationally differentially private. Now consider the following variant of algorithm $\mathcal{B}$, denoted as $\hat{\mathcal{B}}$. The modifications from $\mathcal{B}$ are marked in red.

1. The input is n codewords $w_1, \ldots, w_n \in \{0,1\}^d$.
2. Sample n keys $x_1, \ldots, x_n \sim \mathrm{Gen}(1^\lambda)$.
3. Let $z \leftarrow \mathcal{A}_1(x_1, \ldots, x_n)$.
4. Sample n public parameters $p_1, \ldots, p_n \sim \mathrm{Param}(1^\lambda)$.
5. Let $J \leftarrow J(\mathcal{A}_1, \vec{x}, z, \vec{p}) \subseteq [n]$ be the subset of coordinates guaranteed to exist by Definition 2.1, of size $|J| = n - \overline{\tau}$.
6. For $i \in J$ let $c_i \leftarrow \mathrm{Enc}(x_i, p_i, 0)$.
7. For $i \in [n] \setminus J$ let $c_i \leftarrow \mathrm{Enc}(x_i, p_i, w_i)$.
8. Let $\vec{a} \leftarrow \mathcal{A}_2(z, (p_1, c_1), \ldots, (p_n, c_n))$.
9. Output $\vec{a}$, after rounding its coordinates to $\{0,1\}$.

Remark 3.12. *Observe that algorithm $\hat{\mathcal{B}}$ is not necessarily computationally efficient, since computing $J(\mathcal{A}_1, \vec{x}, z, \vec{p})$ might not be efficient. Nevertheless, as we next show, this still suffices to obtain a contradiction and complete the proof of the lower bound. Specifically, we will show that $\hat{\mathcal{B}}$ is computationally differentially private (w.r.t. the codewords) and that it is a successful adversary to the FPC. This will lead to a contradiction, even if $\hat{\mathcal{B}}$ itself is a non-efficient mechanism.*

We now show that, by the multi-security of the MILR scheme (see Definition 2.1), the outcome distributions of $\mathcal{B}$ and $\hat{\mathcal{B}}$ are computationally indistinguishable. Specifically, we want to show that for every efficient adversary (Q, T), as in Definition 3.5, it holds that

$$|\Pr[T(r, \mathcal{B}(\vec{w})) = 1] - \Pr[T(r, \hat{\mathcal{B}}(\vec{w})) = 1]| \leq \mathrm{negl}(\Gamma(\lambda)).$$

Note that here both expressions are with the same dataset $\vec{w}$ (without the neighboring dataset $\vec{w'}$). To show this, consider the following algorithm, denoted as $\mathcal{W}$, which we view as an adversary to the MILR scheme. This algorithm has only oracle access to encryptions, via an oracle $\mathcal{E}$.

1. The input of $\mathcal{W}$ is n codewords $w_1, \ldots, w_n \in \{0,1\}^d$, an element z (supposedly computed by $\mathcal{A}_1$), and a collection of n public parameters $p_1, \ldots, p_n$.
2. For $i \in [n]$ let $c_i \leftarrow \mathcal{E}(i, w_i)$.
3. Let $\vec{a} \leftarrow \mathcal{A}_2(z, (p_1, c_1), \ldots, (p_n, c_n))$.
4. Output $\vec{a}$, after rounding its coordinates to $\{0,1\}$.

Now by the multi-security of the MILR scheme we have

$$\left| \Pr_{Q,T,\mathcal{B}}[T(r, \mathcal{B}(\vec{w})) = 1] - \Pr_{Q,T,\mathcal{B}}[T(r, \hat{\mathcal{B}}(\vec{w})) = 1] \right|$$

$$= \left| \Pr_{\substack{\vec{x},\vec{p},\mathcal{W},Q,T,\mathrm{Enc} \\ z \leftarrow \mathcal{A}_1(\vec{x}) \\ J \leftarrow J(\mathcal{A}_1,\vec{x},z,\vec{p})}} [Q(r, \mathcal{W}^{\mathcal{E}_1(\vec{x},\vec{p},J,\cdot,\cdot)}(\vec{w}, z, \vec{p})) = 1] \right.$$

$$\left. - \Pr_{\substack{\vec{x},\vec{p},\mathcal{W},Q,T,\mathrm{Enc} \\ z \leftarrow \mathcal{A}_1(\vec{x}) \\ J \leftarrow J(\mathcal{A}_1,\vec{x},z,\vec{p})}} [Q(r, \mathcal{W}^{\mathcal{E}_0(\vec{x},\vec{p},J,\cdot,\cdot)}(\vec{w}, z, \vec{p})) = 1] \right|$$

$$\leq \mathrm{negl}(\Gamma(\lambda)).$$

So we have that $\hat{\mathcal{B}} \equiv_c \mathcal{B}$ and we have that $\mathcal{B}$ is $(2\varepsilon, (e^{\varepsilon}+1)\delta)$-computationally differentially private. Hence, algorithm $\hat{\mathcal{B}}$ is also $(2\varepsilon, (e^{\varepsilon}+1)\delta)$-computationally differentially private (w.r.t. the input codewords). Observe that algorithm $\hat{\mathcal{B}}$ ignores all but $N - |J| = \overline{\tau}$ of the codewords, and furthermore, the choice of which codewords to ignore is independent of the codewords themselves. Now consider the following thought experiment.

1. Sample a codebook $w_0, w_1, \ldots, w_n$ for the fingerprinting code.
2. Run $\hat{\mathcal{B}}$ on $(w_1, \ldots, w_n)$.
3. Run Trace on the outcome of $\hat{\mathcal{B}}$ and return its output.

As $\hat{\mathcal{B}}$ ignores all but $\overline{\tau}$ codewords, by the properties of the FPC, with probability at least $1 - \frac{1}{n^2} \geq \frac{1}{2}$ the outcome of Trace is a coordinate of a codeword that $\hat{\mathcal{B}}$ did not ignore, and in particular, it is a coordinate between 1 and n. Therefore, there must exist a coordinate $i^* \neq 0$ that is output by this thought experiment with probability at least $\frac{1}{2n}$. Now consider the following modified thought experiment.

1. Sample a codebook $w_0, w_1, \ldots, w_n$ for the fingerprinting code.
2. Run $\hat{\mathcal{B}}$ on $(w_1, \ldots, w_{i^*-1}, w_0, w_{i^*+1}, \ldots, w_n)$.
3. Run Trace on the outcome of $\hat{\mathcal{B}}$ and return its output.

As $\hat{\mathcal{B}}$ is computationally differentially private and as Trace is an efficient algorithm, the probability of outputting i^* in this second thought experiment is roughly the same as in the previous thought experiment, specifically, at least

$$e^{-2\varepsilon}\left(\frac{1}{2n} - (e^{\varepsilon}+1)\delta - \mathrm{negl}(\Gamma(\lambda))\right) \geq e^{-2\varepsilon}\left(\frac{1}{4n} - \mathrm{negl}(\Gamma(\lambda))\right) = \Omega\left(\frac{1}{n}\right).$$

However, by the guarantees of the FPC (small probability of false accusation), in the second experiment the probability of outputting i^* should be at most $\frac{1}{n^2}$. This is a contradiction to the existence of algorithm $\mathcal{A}$.

The following corollary follows by instantiating Theorem 3.10 with our MILR scheme, as specified in Theorem 2.3.

Corollary 3.13. *Let $\Omega(\lambda) \leq \Gamma(\lambda) \leq 2^{o(\lambda)}$, and let $d \leq \mathrm{poly}(\Gamma(\lambda))$. If there exists a $\Gamma(\lambda)$-secure encryption scheme against non-uniform adversaries then there exists an MILR scheme such that the corresponding (λ, d)-DA problem requires large space to be solved privately. Specifically, let $n \leq \mathrm{poly}(\Gamma(\lambda))$, let ε be a constant, and let $\delta \leq \frac{1}{4n(e^{\varepsilon}+1)} = \Theta(\frac{1}{n})$. Every $\mathrm{poly}(\Gamma(\lambda))$-time (ε, δ)-CDP algorithm for the (λ, d)-DA problem with sample complexity n must have space complexity $s = \Omega\left(\lambda \cdot \sqrt{\frac{d}{\log n}}\right)$.*

4 Space Hardness for Adaptive Data Analysis (ADA)

Consider a mechanism that first gets as input a sample containing t i.i.d. samples from some underlying (unknown) distribution $\mathcal{D}$, and then answers k *adaptively chosen* statistical queries w.r.t. $\mathcal{D}$. Importantly, the answers must be accurate

Algorithm 1. `AdaptiveGameSpace`$(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \mathcal{B}, s, k)$

1. The adversary $\mathcal{B}$ chooses a distribution $\mathcal{D}$ over a domain $\mathcal{X}$.
2. The mechanism $\mathcal{A}_1$ gets $\mathcal{D}$ and summarizes it into s bits, denoted as z.
3. The mechanism $\mathcal{A}_2$ is instantiated with z.
4. For round $i = 1, 2, \ldots, k$:
   (a) The adversary $\mathcal{B}$ specifies a query $q_i : \mathcal{X} \to \{-1, 0, 1\}$
   (b) The mechanism $\mathcal{A}_2$ obtains q_i and responds with an answer $a_i \in [-1, 1]$
   (c) a_i is given to $\mathcal{A}$
5. The outcome of the game is one if $\exists i$ s.t. $|a_i - \mathbb{E}_{y \sim \mathcal{D}}[q_i(y)]| > 1/10$, and zero otherwise.

w.r.t. the underlying distribution and not just w.r.t. the empirical sample. The challenge here is that as the queries are being chosen adaptively, the interaction might quickly lead to *overfitting*, i.e., result in answers that are only accurate w.r.t. the empirical sample and not w.r.t. the underlying distribution. This fundamental problem, which we refer to as the ADA problem, was introduced by Dwork et al. [15] who connected it to differential privacy and showed that differential privacy can be used as a countermeasure against overfitting. Intuitively, overfitting happens when answers reveal properties that are specific to the input sample, rather than to the general population, and this is exactly what differential privacy aims to protect against.

Hardt, Steinke, and Ullman [24,34] showed negative results for the ADA problem. Specifically, they showed that given t samples, it is computationally hard to answer more than $k = O(t^2)$ adaptive queries. We show that the hardness of the ADA problem is actually more fundamental; it is, in fact, a result of a space bottleneck rather than a sampling bottleneck. Informally, we show that the same hardness result continues to hold even if in the preprocessing stage the mechanism is given the full description of the underlying distribution $\mathcal{D}$, and is then required to store only a limited amount of information about it (an amount that equals the representation length of t samples from $\mathcal{D}$). So it is not that the mechanism did not get enough information about $\mathcal{D}$; it is just that it cannot shrink this information in a way that supports t^2 adaptive queries. This generalizes the negative results of [24,34], as sampling t points from $\mathcal{D}$ is just one particular way of trying to store information about $\mathcal{D}$.

Consider `AdaptiveGameSpace`, where the mechanism initially gets the full description of the underlying distribution, but it must shrink it into an s-bit summary z. To emphasize that the mechanism does not have additional access to the underlying distribution, we think about it as two mechanisms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1$ computes the summary z and where $\mathcal{A}_2$ answers queries given z. We consider $s = |z|$ as the space complexity of such a mechanism $\mathcal{A}$.

Our main theorem in the context of the ADA problem is the following.

Theorem 4.1. *Let* $\Omega(\lambda) \leq \Gamma(\lambda) \leq 2^{o(\lambda)}$*, and let* $k \leq \text{poly}(\Gamma(\lambda))$*. If there exists a* $\Gamma(\lambda)$*-secure encryption scheme against non-uniform adversaries then there exists a* $\text{poly}(\Gamma(\lambda))$*-time adversary* $\mathcal{B}$ *such that the following holds. Let* $\mathcal{A}=(\mathcal{A}_1, \mathcal{A}_2)$ *be a* $\text{poly}(\Gamma(\lambda))$*-time mechanism with space complexity* $s \leq O\left(\lambda \cdot \sqrt{k}\right)$ *(small enough). Then,* $\Pr[\texttt{AdaptiveGameSpace}(\mathcal{A}, \mathcal{B}, s, k) = 1] > \frac{2}{3}$.

Furthermore, the underlying distribution defined by the adversary $\mathcal{B}$ *can be fully described using* $O(\sqrt{k} \cdot \lambda)$ *bits, it is sampleable in* $\text{poly}(\Gamma(\lambda))$*-time, and elements sampled from this distribution can be represented using* $O(\lambda + \log(k))$ *bits.*

In a sense, the "furtheremore" part of the theorem shows that the distribution chosen by our adversary is not too complex. Specifically, our negative result continues to hold even if the space of the mechanism is *linear* in the full description length of the underlying distribution (in a way that allows for efficiently sampling it). If the space of the mechanism was just a constant times bigger, it could store the full description of the underlying distribution and answer an unbounded number of adaptive queries. The formal proof of Theorem 4.1 is deferred to the full version of this work. Here we only provide an informal (and overly simplified) proof sketch.

4.1 Informal Proof Sketch

Let k denote the number of queries that the adversary makes. Our task is to show that there is an adversary that fails every efficient mechanism $\mathcal{A}_{\text{space}}$ that plays in `AdaptiveGameSpace`, provided that it uses space $s \ll \sqrt{k}$. What we know from [24,34] is that there is an adversary $\mathcal{B}_{\text{sample}}$ that fails every efficient mechanism that plays in the standard ADA game (the game specified in Fig. 1), provided that its sample complexity is $t \ll \sqrt{k}$. We design an adversary $\mathcal{B}_{\text{space}}$ that plays in `AdaptiveGameSpace` in a way that emulates $\mathcal{B}_{\text{sample}}$. We now elaborate on the key points in the construction of $\mathcal{B}_{\text{space}}$, and their connection to $\mathcal{B}_{\text{sample}}$.

Recall that both games begin with the adversary specifying the underlying distribution. A useful fact about the adversary $\mathcal{B}_{\text{sample}}$ (from [24,34]) is that the distribution it specifies is uniform on a small set of points of size $n = \Theta(t)$ (these n points are unknown to the mechanism that $\mathcal{B}_{\text{sample}}$ plays against). Our adversary, $\mathcal{B}_{\text{space}}$, first samples n independent keys $(x_1, \ldots, x_n)$ from our MILR scheme, and then defines the target distribution $\mathcal{D}_{\text{space}}$ to be uniform over the set $\{(j, x_j)\}_{j \in [n]}$. Recall that in `AdaptiveGameSpace` this target distribution is given to the mechanism $\mathcal{A}_{\text{space}}$, who must shrink it into a summary z containing s bits. After this stage, by the security of our MILR scheme, there should exist a large set $J \subseteq [n]$ of size $|J| = n - \overline{\tau}$ corresponding to keys uncompromised by $\mathcal{A}_{\text{space}}$. Denote $I = [n] \setminus J$.

Our adversary $\mathcal{B}_{\text{space}}$ now emulates $\mathcal{B}_{\text{sample}}$ as follows. First, let $\mathcal{D}_{\text{sample}}$ denote the target distribution chosen by $\mathcal{B}_{\text{sample}}$, and let $m_1, \ldots, m_n$ denote its support. Our adversary then samples n public parameters $p_1, \ldots, p_n$, and encrypts every point in the support m_j using its corresponding key and public

parameter. Specifically, $c_j \leftarrow \mathrm{Enc}(x_j, p_j, m_j)$. % This is an over-simplification. For technical reasons, the actual construction is somewhat different.

Now, for every query q specified by $\mathcal{B}_{\mathrm{sample}}$, our adversary outputs the query f_q defined by $f_q(j, x) = q(\mathrm{Dec}(x, p_j, c_j))$. Our adversary then obtains an answer a from the mechanism $\mathcal{A}_{\mathrm{space}}$, and feeds a to $\mathcal{B}_{\mathrm{sample}}$. Observe that the "true" value of f_q w.r.t. $\mathcal{D}_{\mathrm{space}}$ is the same as the "true" value of q w.r.t. $\mathcal{D}_{\mathrm{sample}}$. Therefore, if $\mathcal{A}_{\mathrm{space}}$ maintains accuracy in this game against our adversary $\mathcal{B}_{\mathrm{space}}$, then in the emulation that $\mathcal{B}_{\mathrm{space}}$ runs internally we have that $\mathcal{B}_{\mathrm{space}}$ maintains utility against $\mathcal{B}_{\mathrm{sample}}$. Intuitively, we would like to say that this leads to a contradiction, since $\mathcal{B}_{\mathrm{sample}}$ fails every efficient mechanism it plays against. But this is not accurate, because $\mathcal{B}_{\mathrm{space}}$ saw the full description of the target distribution $\mathcal{D}_{\mathrm{sample}}$, and $\mathcal{B}_{\mathrm{sample}}$ only fools mechanisms that get to see at most t *samples* from this target distribution.

To overcome this, we consider the following modified variant of our adversary, called $\widehat{\mathcal{B}}_{\mathrm{space}}$. The modification is that $\widehat{\mathcal{B}}_{\mathrm{space}}$ does not get to see the full description of $\mathcal{D}_{\mathrm{sample}}$. Instead it only gets to see points from the support of $\mathcal{D}_{\mathrm{sample}}$ that correspond to indices in the set $I = [n] \setminus J$. Then, when generating the ciphertexts c_j, the modified adversary $\widehat{\mathcal{B}}_{\mathrm{space}}$ encrypts zeroes instead of points m_j which it is missing. By the security of our MILR scheme, the mechanism $\mathcal{A}_{\mathrm{space}}$ cannot notice this modification, and hence, assuming that it maintains accuracy against our original adversary $\mathcal{B}_{\mathrm{space}}$ then it also maintains accuracy against our modified adversary $\widehat{\mathcal{B}}_{\mathrm{space}}$. As before, this means that $\widehat{\mathcal{B}}_{\mathrm{space}}$ maintains accuracy against the emulated $\mathcal{B}_{\mathrm{sample}}$. Intuitively, this leads to a contradiction, as $\widehat{\mathcal{B}}_{\mathrm{space}}$ is using only $\overline{\tau} \leq t$ points from the target distribution $\mathcal{D}_{\mathrm{sample}}$.

We stress that this proof sketch is over-simplified and inaccurate. In particular, the following two technical issues need to be addressed: (1) It is true that $\widehat{\mathcal{B}}_{\mathrm{space}}$ uses only $\overline{\tau}$ points from the support of the target distribution $\mathcal{D}_{\mathrm{sample}}$, but these points are not necessarily *sampled* from $\mathcal{D}_{\mathrm{sample}}$; and (2) The modified adversary $\widehat{\mathcal{B}}_{\mathrm{space}}$ is not computationally efficient because computing the set J is not efficient. We address these issues, and other informalities made herein, in the full version of this work.

5 Construction of an MILR Scheme from a Semantically Secure Encryption Scheme

Construction. Let $\lambda' \leq \lambda$ be such that $\lambda = \mathrm{poly}(\lambda')$. Given an encryption scheme $\Pi' = (\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ such that $\mathrm{Gen}'(1^{\lambda'})$ outputs a key uniformly distributed on $\{0,1\}^{\lambda'}$ (i.e., $x' \leftarrow_R \{0,1\}^{\lambda'}$), we construct an MILR scheme $\Pi = (\mathrm{Gen}, \mathrm{Param}, \mathrm{Enc}, \mathrm{Dec})$ as follows:

- Gen: On input 1^λ, return $x \leftarrow_R \{0,1\}^\lambda$.
- Param: On input 1^λ, let $\mathcal{G}$ be a family of universal hash functions with domain $\{0,1\}^\lambda$ and range $\{0,1\}^{\lambda'}$. Return (a description of) $g \leftarrow_R \mathcal{G}$.
- Enc: On input (x,p,m), parse $g := p$ (as a description of a hash function), let $x' = g(x)$ and return $\mathrm{Enc}'(x',m)$.
- Dec: On input (x,p,c), parse $g := p$, let $x' = g(x)$ and return $\mathrm{Dec}'(x',c)$.

Using a standard construction of a universal hash function family, all the algorithms run in time polynomial in λ. Moreover, if $c \leftarrow \mathrm{Enc}(x,p,m)$, then $\mathrm{Dec}(x,p,c) = m$ with probability 1 (as this holds for Enc' and Dec').

The following two theorems (corresponding to the two security properties in Definition 2.1) establish the security of Π and prove Theorem 2.3.

Theorem 5.1 (Multi semantic security). *Let $\Omega(\lambda) \leq \Gamma(\lambda) \leq 2^{o(\lambda)}$ and $\lambda' = 0.1\lambda$. If Π' is $\Gamma(\lambda')$-secure against uniform (resp. non-uniform) adversaries, then Π is $\Gamma(\lambda)$-secure against uniform (resp. non-uniform) adversaries.*

Theorem 5.2 (Multi-security against bounded preprocessing adversaries). *Let $\Omega(\lambda) \leq \Gamma(\lambda) \leq 2^{o(\lambda)}$, $\lambda' = 0.1\lambda$ (as in Theorem 5.1). If Π' is $\Gamma(\lambda')$-secure against non-uniform adversaries then Π is* $(\Gamma(\lambda), \overline{\tau})$-secure *against space bounded non-uniform preprocessing adversaries, where $\overline{\tau}(\lambda, s) = \frac{2s}{\lambda} + 4$.*

Remark 5.3. *Since $\Gamma(\lambda) \leq 2^{o(\lambda)}$ and $\lambda' = 0.1\lambda$, then* $\mathrm{poly}(\Gamma(\lambda')) = \mathrm{poly}(\Gamma(\lambda))$. *Therefore, for the sake of simplicity, we analyze the runtime and advantage of all adversaries (including those that run against Π') as functions of λ.*

The proof of Theorem 5.1 is given in the full version of this work. We prove Theorem 5.2 in Sect. 6.

6 Multi-security Against a Bounded Preprocessing Adversary

In this section we prove Theorem 5.2. The proof requires specific definitions and notation, defined below.

6.1 Preliminaries

Notation. Given a sequence of elements $X = (X_1, \ldots, X_n)$ and a subset $I \subseteq [n]$, we denote by X_I the sequence composed of elements with coordinates in I.

For a random variable X, denote its min-entropy by $H_\infty(X)$. For random variables X, Y with the same range, denote by $\Delta(X,Y)$ the statistical distance of their distributions. We say that X and Y are γ-close if $\Delta(X,Y) \leq \gamma$. We use the notation $X \leftarrow_R \mathcal{X}$ to indicate that the random variable X is chosen uniformly at random from the set $\mathcal{X}$.

Dense and Bit-Fixing Sources. We will use the following definition (see [12, Definition 1]).

Definition 6.1. *An $(n, 2^\lambda)$-source is a random variable X with range $(\{0,1\}^\lambda)^n$. A source is called*

- *$(1-\delta)$-dense if for every subset $I \subseteq [n]$, $H_\infty(X_I) \geq (1-\delta) \cdot |I| \cdot \lambda$,*
- *$(k, 1-\delta)$-dense if it is fixed on at most k coordinates and is $(1-\delta)$-dense on the rest,*
- *k-bit-fixing if it is fixed on at most k coordinates and uniform on the rest.*

Namely, the min-entropy of every subset of entries of a $(1-\delta)$-dense source is at most a fraction of δ less than what it would be for a uniformly random one.

6.2 Key Leakage Lemma

Let $X = (X_1, \ldots, X_n) \in (\{0,1\}^\lambda)^n$ be a random variable for n keys of Π chosen independently and uniformly at random. Let $Z := F(X)$ be a random variable for the leakage of the adversary. For $z \in \{0,1\}^s$, let X_z be the random variable chosen from the distribution of X conditioned on $F(X) = z$.

We denote $G := (G_1, \ldots, G_n)$ and $G(X) := (G_1(X_1), \ldots, G_n(X_n))$ the random variable for the hash functions (public parameters) of Π. We will use similar notation for sequences of different lengths (which will be clear from the context).

The proof of Theorem 5.2 is based on the lemma below (proved in Sect. 6.4), which analyzes the joint distribution $(G, Z, G(X))$.

Lemma 6.2. *Let $F : (\{0,1\}^\lambda)^n \to \{0,1\}^s$ be an arbitrary function, $X = (X_1, \ldots, X_n) \leftarrow_R (\{0,1\}^\lambda)^n$ and denote $Z := F(X)$. Let $\mathcal{G}$ be a family of universal hash functions with domain $\{0,1\}^\lambda$ and range $\{0,1\}^{\lambda'}$ and let $G \leftarrow_R (\mathcal{G})^n$. Let $\delta > 0, \gamma > 0, s' > s$ be parameters such that $(1-\delta)\lambda > \lambda' + \log n + 1$.*

Then, there exists a family $V_{G,Z} = \{V_{\vec{g},z}\}_{\vec{g} \in (\mathcal{G})^n, z \in \{0,1\}^s}$ of convex combinations $V_{\vec{g},z}$ of k-bit-fixing $(n, 2^{\lambda'})$-sources for $k = \frac{s' + \log 1/\gamma}{\delta \cdot \lambda}$ such that

$$\Delta[(G, Z, G(X)), (G, Z, V_{G,Z})] \leq \sqrt{2^{-(1-\delta)\lambda + \lambda' + \log n}} + \gamma + 2^{s-s'}.$$

We obtain the following corollary (which implies the parameters of Theorem 5.2).

Corollary 6.3. *In the setting of Lemma 6.2, assuming $n < 2^{0.15\lambda}$ and sufficiently large λ, the parameters $s' = s + \lambda, \lambda' = 0.1\lambda, \delta = 0.5, \gamma = 2^{-\lambda}$ give*

$$\Delta[(G, Z, G(X)), (G, Z, V_{G,Z})] \leq 2^{-0.1\lambda}, \qquad \text{and} \qquad k = \frac{2s}{\lambda} + 4.$$

Proof. Set $s' = s + \lambda$, $\lambda' = 0.1\lambda$, $\delta = 0.5, \gamma = 2^{-\lambda}$. Then, for sufficiently large λ, $(1-\delta)\lambda = 0.5\lambda > 0.1\lambda + 0.15\lambda + 1 > \lambda' + \log n + 1$ (and the condition of Lemma 6.2 holds). We therefore have (for sufficiently large λ): $\Delta[(G, Z, G(X)), (G, Z, V_{G,Z})] \leq \sqrt{2^{-(1-\delta)\lambda+\lambda'+\log n}} + \gamma + 2^{s-s'} = \sqrt{2^{-0.4\lambda+\log n}} + 2^{-\lambda} + 2^{-\lambda} \leq 2^{-0.1\lambda}$, and $k = \frac{s'+\log 1/\gamma}{\delta\cdot\lambda} = \frac{s+\lambda+\lambda}{0.5\cdot\lambda} = \frac{2s}{\lambda} + 4$.

6.3 The Proof of Theorem 5.2

Using Lemma 6.2 to Prove Theorem 5.2. Before proving Theorem 5.2, we explain why Lemma 6.2 is needed and how it used in the proof.

It is easy to prove some weaker statements than Lemma 6.2, but these do not seem to be sufficient for building the MILR scheme (i.e., proving Theorem 5.2). For example, one can easily prove that with high probability, given the leakage z and hash functions $\vec{g}$, there is a large subset of (hashed) keys such that each one of them is almost uniformly distributed. However, the adversary could have knowledge of various relations between the keys of this subset and it is not clear how to prove security without making assumptions about the resistance of the encryption scheme against related-key attacks.

Moreover, consider a stronger statement, which asserts that with high probability, given the leakage z and hash functions $\vec{g}$, there is a large subset of (hashed) keys that are jointly uniformly distributed. We claim that even this stronger statement may not be sufficient to prove security, since it does not consider the remaining keys outside of the subset. In particular, consider a scenario in which the adversary is able to recover some weak keys outside of the subset. Given this extra knowledge and the leakage z, the original subset of keys may no longer be distributed uniformly (and may suffer from a significant entropy loss).

Lemma 6.2 essentially asserts that there is a subset of keys that is almost jointly uniformly distributed even if we give the adversary $z, \vec{g}$ and all the remaining keys. More specifically, given the hash functions $\vec{g}$ and the leakage z, according to the lemma, the distribution of the hashed keys $\vec{g}(X)$ is (close to) a convex combinations $V_{\vec{g},z}$ of k-bit-fixing sources. In the proof of Theorem 5.2 we will fix such a k-bit-fixing source by giving the adversary k hashed keys (we will do this carefully, making sure that the adversary's advantage does not change significantly). Since the remaining hashed keys are uniformly distributed from the adversary's view, security with respect to these keys follows from the semantic security of the underlying encryption scheme.

Proof (Proof of Theorem 5.2). Fix a preprecessing procedure F and let λ be sufficiently large, $n < 2^{0.15\lambda}$. By Lemma 6.2 (with parameters set in Corollary 6.3), there exists a family $V_{G,Z}$ of convex combinations $V_{\vec{g},z}$ of k-bit-fixing $(n, 2^{\lambda'})$-sources for $k = \frac{2s}{\lambda} + 4$ such that $\Delta[(G, Z, G(X)), (G, Z, V_{G,Z})] \leq 2^{-0.1\lambda}$.

Sampling the Index Set J and Simplifying the Distribution. We first define how the oracles $\mathcal{E}_0$ and $\mathcal{E}_1$ in Definition 2.1 sample J. The random variable J is naturally defined when sampling the random variables $(G, Z, V_{G,Z})$: given $\vec{g} \in (\mathcal{G})^n, z \in \{0,1\}^s$, sample a k-bit-fixing source in the convex combination $V_{\vec{g},z}$

(according to its weight) and let J be the set of (at least) $n-k$ indices that are not fixed. This defines a joint distribution on the random variables $(G, Z, V_{G,Z}, J)$. Another way to sample from this distribution is to first sample the variables $(G, Z, V_{G,Z})$ and then sample J according to its marginal distribution. This defines a randomized procedure for sampling J. Although the oracles do not sample $(G, Z, V_{G,Z})$, we reuse the same sampling procedure for sampling J given the sample $(\vec{g}, z, \vec{g}(\vec{x}))$ (if the sample $(\vec{g}, z, \vec{g}(\vec{x}))$ is not in the support of the distribution of $(G, Z, V_{G,Z})$, define $J = [n]$).

Consider a poly$(\Gamma(\lambda))$-time algorithm $\mathcal{B}$. As encryption queries of $\mathcal{B}$ to $\mathcal{E}_0$ and $\mathcal{E}_1$ are answered with the hash keys $\vec{g}(\vec{x})$, then given $\vec{g}(\vec{x})$, the interaction of $\mathcal{B}$ with $\mathcal{E}_0$ and $\mathcal{E}_1$ no longer depends on $\vec{x}$. Therefore, for $t = 0, 1$ we define $\mathcal{E}_t^{(1)}(\vec{g}(\vec{x}), \vec{g}, J, \cdot, \cdot)$ that simulates the interaction of $\mathcal{E}_t(\vec{x}, \vec{g}, J, \cdot, \cdot)$ with $\mathcal{B}$. Instead of sampling $\vec{x}$, the oracles directly sample $(\vec{g}, z, \vec{g}(\vec{x}), J)$ according to their joint distribution before the interaction with $\mathcal{B}$. To simplify notation, we denote this joint distribution by $\mathcal{D}_1$. Denote

$$\mathrm{Adv}_{\mathcal{B}}(\lambda) = \left| \Pr_{\substack{\mathcal{B},\mathrm{Enc} \\ (\vec{g},z,\vec{g}(\vec{x}),J)\leftarrow\mathcal{D}_1}} \left[\mathcal{B}^{\mathcal{E}_0^{(1)}(\vec{g}(\vec{x}),\vec{g},J,\cdot,\cdot)}(z,\vec{g}) = 1\right] - \Pr_{\substack{\mathcal{B},\mathrm{Enc} \\ (\vec{g},z,\vec{g}(\vec{x}),J)\leftarrow\mathcal{D}_1}} \left[\mathcal{B}^{\mathcal{E}_1^{(1)}(\vec{g}(\vec{x}),\vec{g},J,\cdot,\cdot)}(z,\vec{g}) = 1\right] \right|.$$

It remains to prove that $\mathrm{Adv}_{\mathcal{B}}(\lambda) \leq \mathrm{negl}(\Gamma(\lambda))$.

Using Lemma 6.2 to switch to a family of convex combinations of bit-fixing sources. We have

$$\Delta[(G, Z, G(X), J), (G, Z, V_{G,Z}, J)] \leq \Delta[(G, Z, G(X)), (G, Z, V_{G,Z})] \leq 2^{-0.1\lambda},$$

where the first inequality follows by the data processing inequality, since J is computed by applying the same function to the three variables of both distributions, and the second inequality is by Corollary 6.3. Hence, for $t = 0, 1$ we replace $\mathcal{E}_t^{(1)}$ that samples from $\mathcal{D}_1$ with $\mathcal{E}_t^{(2)}$ that samples from the joint distribution of $(G, Z, V_{G,Z}, J)$, which we denote by $\mathcal{D}_2$. Since $\mathcal{B}$ and Enc use independent randomness, by the triangle inequality, the total penalty is at most $2 \cdot 2^{-0.1\lambda}$, namely

$$\left| \Pr_{\substack{\mathcal{B},\mathrm{Enc} \\ (\vec{g},z,\vec{y},J)\leftarrow\mathcal{D}_2}} \left[\mathcal{B}^{\mathcal{E}_0^{(2)}(\vec{y},\vec{g},J,\cdot,\cdot)}(z,\vec{g}) = 1\right] - \Pr_{\substack{\mathcal{B},\mathrm{Enc} \\ (\vec{g},z,\vec{y},J)\leftarrow\mathcal{D}_2}} \left[\mathcal{B}^{\mathcal{E}_1^{(2)}(\vec{y},\vec{g},J,\cdot,\cdot)}(z,\vec{g}) = 1\right] \right| \geq \mathrm{Adv}_{\mathcal{B}}(\lambda) - 2^{-0.1\lambda+1},$$

where we denote a sample from $\mathcal{D}_2$ by $(\vec{g}, z, \vec{y}, J)$.

Giving the Adversary Additional Input. Consider a (potentially) more powerful poly$(\Gamma(\lambda))$-time algorithm $\mathcal{B}_1$ against Π whose input consists of $(z, \vec{g}, J, \vec{y}_{\overline{J}})$, where $\overline{J} = [n] \backslash J$. Namely, in addition to $(z, \vec{g})$ the input also consists of J, as

well as the hashed keys $\vec{y}_{\overline{J}} \in \left(\{0,1\}^{\lambda'}\right)^{n-|J|}$ (note that these parameters define a $|\overline{J}|$-bit-fixing source). We denote $in = (z, \vec{g}, J, \vec{y}_{\overline{J}})$, and

$$\mathrm{Adv}_{\mathcal{B}_1}(\lambda) = \left| \Pr_{\substack{\mathcal{B}_1,\mathrm{Enc} \\ (\vec{g},z,\vec{y},J)\leftarrow\mathcal{D}_2}} \left[\mathcal{B}_1^{\mathcal{E}_0^{(2)}(\vec{y},\vec{g},J,\cdot,\cdot)}(in) = 1\right] - \Pr_{\substack{\mathcal{B}_1,\mathrm{Enc} \\ (\vec{g},z,\vec{y},J)\leftarrow\mathcal{D}_2}} \left[\mathcal{B}_1^{\mathcal{E}_1^{(2)}(\vec{y},\vec{g},J,\cdot,\cdot)}(in) = 1\right] \right|.$$

Next, we prove that for any such $\mathrm{poly}(\Gamma(\lambda))$-time algorithm $\mathcal{B}_1$, $\mathrm{Adv}_{\mathcal{B}_1}(\lambda) \leq \mathrm{negl}(\Gamma(\lambda))$. As any algorithm $\mathcal{B}$ with input $(z, \vec{g})$ can be simulated by an algorithm $\mathcal{B}_1$ with input in and similar runtime, this implies that $\mathrm{Adv}_{\mathcal{B}}(\lambda) - 2^{-0.1\lambda+1} \leq \mathrm{negl}(\Gamma(\lambda))$ and hence $\mathrm{Adv}_{\mathcal{B}}(\lambda) \leq \mathrm{negl}(\Gamma(\lambda))$, concluding the proof.

Fixing the Adversary's Input. Since both $\mathcal{E}_0^{(2)}$ and $\mathcal{E}_1^{(2)}$ sample the input of $\mathcal{B}_1$ from the same distribution, by an averaging argument, there exists an input $in^* = in^*_\lambda = (z^*, \vec{g}^*, J^*, \vec{y}^*_{\overline{J^*}})$ such that the advantage of $\mathcal{B}_1$ remains at least as large when fixing the input to in^* and sampling from $\mathcal{D}_2$ conditioned on in^*. Note that given in^* sampling from $\mathcal{D}_2$ reduces to sampling from the $|\overline{J^*}|$-bit-fixing source defined by $(\overline{J^*}, \vec{y}^*_{\overline{J^*}})$, i.e., selecting $\vec{w} \leftarrow_R (\{0,1\}^{\lambda'})^{|J^*|}$. Therefore,

$$\left| \Pr_{\substack{\mathcal{B}_1,\mathrm{Enc} \\ \vec{w}\leftarrow_R(\{0,1\}^{\lambda'})^{|J^*|}}} \left[\mathcal{B}_1^{\mathcal{E}_0^{(2)}(in^*,\vec{w},\cdot,\cdot)}(in^*) = 1\right] - \Pr_{\substack{\mathcal{B}_1,\mathrm{Enc} \\ \vec{w}\leftarrow_R(\{0,1\}^{\lambda'})^{|J^*|}}} \left[\mathcal{B}_1^{\mathcal{E}_1^{(2)}(in^*,\vec{w},\cdot,\cdot)}(in^*) = 1\right] \right| \geq \mathrm{Adv}_{\mathcal{B}_1}(\lambda).$$

Reducing the Security of Π with Preprocessing from the (Multi-instance) Security of Π'. We now use $\mathcal{B}_1$ to define a non-uniform $\mathrm{poly}(\Gamma(\lambda))$-time adversary $\mathcal{B}_2$ (with no preprocessing) that runs against $|J^*|$ instances of Π' and has advantage at least $\mathrm{Adv}_{\mathcal{B}_1}(\lambda)$. By the semantic security of Π' and a hybrid argument (similarly to the proof of Theorem 5.1), this implies that $\mathrm{Adv}_{\mathcal{B}_1}(\lambda) \leq \mathrm{negl}(\Gamma(\lambda))$, concluding the proof.

The adversary $\mathcal{B}_2$ is given in Algorithm 2. Note that $\mathcal{B}_2$ perfectly simulates the oracles of $\mathcal{B}_1$ given the input in^*, and hence its advantage is at least $\mathrm{Adv}_{\mathcal{B}_1}(\lambda)$ as claimed. Finally, it runs in time $\mathrm{poly}(\Gamma(\lambda))$.

6.4 Proof of Lemma 6.2

Proof Overview. We first prove in Lemma 6.4 that $G = (G_1, \ldots, G_t)$ (for some $t \in [n]$) is a good extractor, assuming its input $Y = (Y_1, \ldots, Y_t)$ is $(1-\delta)$-dense, namely, it has sufficient min-entropy for each subset of coordinates (see Lemma 6.4 for the exact statement). Specifically, we prove that $(G, G(Y))$ is statistically close to (G, U), where U is uniformly distributed over $(\{0,1\}^{\lambda'})^t$. The proof is by a variant of the leftover hash lemma [25] where a sequence of hash functions $(G_1, \ldots, G_t)$ are applied locally to each block of the input (instead of

Algorithm 2. $\mathcal{B}_2^{\mathcal{E}_{(\cdot)}^{(3)}(\vec{x}',[|J^*|],\cdot,\cdot)}()$

Setting: $\mathcal{B}_2$ is a non-uniform adversary that runs against $|J^*|$ instances of Π' (defined by $\mathcal{E}_{(\cdot)}^{(3)}$). It gets $in^* = in_\lambda^* = (z^*, \vec{y}^*, J^*, \vec{y^*}_{\overline{J^*}})$ as advice. $\mathcal{B}_2$ has access to $\mathcal{B}_1^{\mathcal{E}_{(\cdot)}^{(2)}(in^*,\vec{w},\cdot,\cdot)}(in^*)$, which runs against Π.

1. $\mathcal{B}_2$ gives in^* to $\mathcal{B}_1$ as input.
2. $\mathcal{B}_2$ answers each query (j, m) of $\mathcal{B}_1$ as follows:
   - If $j \in \overline{J^*}$, $\mathcal{B}_2$ uses the advice string in^* (which contains $\vec{y}_j^*$) to compute the answer $\mathrm{Enc}'(\vec{y}_j^*, m)$ and gives it to $\mathcal{B}_1$.
   - If $j \in J^*$, $\mathcal{B}_2$ translates the query (j, m) to (j', m), where $j' \in [|J^*|]$ is obtained by mapping j to J^* (ignoring indices in $\overline{J^*}$). $\mathcal{B}_2$ then queries its oracle with (j', m) and forwards the answer to $\mathcal{B}_1$.
3. $\mathcal{B}_2$ outputs the same output as $\mathcal{B}_1$.

applying a single hash function to the entire input). We note that a related lemma was proved in [22, Lem. 13] in a different setting of communication complexity. Our variant is applicable to a different (mostly wider) range of parameters (such as various values of δ and the number of bits extracted, $t \cdot \lambda'$) that is relevant in our setting. Additional (somewhat less related) results were presented in [13,14].

The remainder of the proof is deferred to the full version of this work, and is somewhat similar to [12, Lem. 1].

Block-Wise Extraction from Dense Sources.

Lemma 6.4 *Let $Y = (Y_1, \ldots, Y_t) \in (\{0,1\}^\lambda)^t$ be a $(t, 2^\lambda)$-source that is $(1-\delta)$-dense for $0 < \delta < 1$. Let $\mathcal{G}$ be a family of universal hash functions with domain $\{0,1\}^\lambda$ and range $\{0,1\}^{\lambda'}$. Then, for $G \leftarrow_R (\mathcal{G})^t$ and $U \leftarrow_R (\{0,1\}^{\lambda'})^t$,*

$$\Delta[(G, G(Y)), (G, U)] \leq \sqrt{2^{-(1-\delta)\lambda+\lambda'+\log t}},$$

assuming that $(1-\delta)\lambda > \lambda' + \log t + 1$.

Proof. Let $d := \log |\mathcal{G}|$. For a random variable Q, and Q' an independent copy of Q, we denote by $Col[Q] = \Pr[Q = Q']$ the collision probability of Q. We have

$$\begin{aligned} Col[(G, G(Y))] &= \Pr_{G,Y,G',Y'}[(G, G(Y')) = (G', G'(Y'))] \\ &= \Pr_{G,G'}[G = G'] \cdot \Pr_{G,Y,Y'}[G(Y) = G(Y')] = 2^{-t\cdot d} \cdot \Pr_{G,Y,Y'}[G(Y) = G(Y')]. \end{aligned} \tag{1}$$

For sequences $Y_1, \ldots, Y_t, Y_1', \ldots, Y_t'$, define $C = |\{i \mid Y_i = Y_i'\}|$. We now upper bound the expression $\Pr[C = c]$.

Recall that Y is a $(1-\delta)$-dense source, i.e., for every subset $I \subseteq [t]$, $H_\infty(Y_I) \geq (1-\delta)\cdot|I|\cdot\lambda$. Fix a subset $I \subseteq [t]$ such that $|I| = c$. Then,

$$\Pr[Y_I = Y'_I] = \sum_{y_I \in (\{0,1\}^\lambda)^c} (\Pr[Y_I = y_I])^2$$
$$\leq \max_{y_I}\{\Pr[Y_I = y_I]\} \cdot \sum_{y_I \in (\{0,1\}^\lambda)^c} \Pr[Y_I = y_I] \leq 2^{-(1-\delta)\cdot c\cdot\lambda}.$$

Therefore,

$$\Pr[C = c] \leq \sum_{\{I\subseteq[t]||I|=c\}} \Pr[Y_I = Y'_I] \leq \binom{t}{c}\cdot 2^{-(1-\delta)\cdot c\cdot\lambda}$$
$$\leq t^c \cdot 2^{-(1-\delta)\cdot c\cdot\lambda} = 2^{c\cdot(-(1-\delta)\lambda+\log t)}. \tag{2}$$

We have

$$\Pr_{G,Y,Y'}[G(Y) = G(Y')] = \sum_{c=0}^{t} \Pr_{Y,Y'}[C = c] \cdot \Pr_{G,Y,Y'}[G(Y) = G(Y') \mid C = c].$$

For each coordinate i such that $Y_i \neq Y'_i$, $\Pr_{G_i}(G_i(Y_i) = G_i(Y'_i)) = 2^{-\lambda'}$ as G_i is selected uniformly from a family of universal hash functions. Since $G = (G_1, \ldots, G_t)$ contains t independent copies selected uniformly from $\mathcal{G}$,

$$\Pr_G[G(Y) = G(Y') \mid C = c] = 2^{-\lambda'\cdot(t-c)}.$$

Hence, using (2) we obtain

$$\Pr_{G,Y,Y'}[G(Y) = G(Y')] = \sum_{c=0}^{t} \Pr[C = c]\cdot 2^{-\lambda'\cdot(t-c)}$$
$$\leq \sum_{c=0}^{t} 2^{c\cdot(-(1-\delta)\lambda+\log t)}\cdot 2^{-\lambda'\cdot(t-c)} = 2^{-\lambda'\cdot t}\cdot\sum_{c=0}^{t} 2^{-c\cdot((1-\delta)\lambda-\lambda'-\log t)}$$
$$= 2^{-\lambda'\cdot t}\cdot(1+\sum_{c=1}^{t} 2^{-c\cdot((1-\delta)\lambda-\lambda'-\log t)}) \leq 2^{-\lambda'\cdot t}\cdot(1+2^{-(1-\delta)\lambda+\lambda'+\log t+1}),$$

where the last inequality uses the assumption that $(1-\delta)\lambda > \lambda' + \log t + 1$.

Treating distributions as vectors over $\{0,1\}^{t\cdot d+t\cdot\lambda'}$ (and abusing notation), we plug the above expression into (1) and deduce

$$\|(G, G(Y)) - (G, U)\|_2^2 = Col[(G, G(Y))] - 2^{-t\cdot d-t\cdot\lambda'} \leq$$
$$2^{-t\cdot d-t\cdot\lambda'}\cdot(1+2^{-(1-\delta)\lambda+\lambda'+\log t+1}) - 2^{-t\cdot d-t\cdot\lambda'} = 2^{-t\cdot d-t\cdot n'-(1-\delta)\lambda+\lambda'+\log t+1}.$$

Finally, using the Cauchy-Schwarz inequality, we conclude

$$\Delta[(G, G(Y)), (G, U)] \leq 1/2 \cdot \sqrt{2^{t \cdot d + t \cdot \lambda'}} \cdot \|(G, G(Y)) - (G, U)\|_2$$
$$\leq 1/2 \cdot \sqrt{2^{t \cdot d + t \cdot \lambda'}} \cdot \sqrt{2^{-t \cdot d - \lambda' \cdot t - (1-\delta)\lambda + \lambda' + \log t + 1}} < \sqrt{2^{-(1-\delta)\lambda + \lambda' + \log t}}.$$

Acknowledgements. Itai Dinur was partially supported by the Israel Science Foundation (grant 1903/20) and by the European Research Council under the ERC starting grant agreement no. 757731 (LightCrypt). Uri Stemmer was partially supported by the Israel Science Foundation (grant 1871/19) and by Len Blavatnik and the Blavatnik Family foundation. Work done in part while David P. Woodruff was visiting Google Research and Samson Zhou was at Carnegie Mellon University. They were also supported by a Simons Investigator Award and by the National Science Foundation under Grant No. CCF-1815840.

References

1. Alon, N., Livni, R., Malliaris, M., Moran, S.: Private PAC learning implies finite littlestone dimension. In: STOC, pp. 852–860. ACM (2019). https://doi.org/10.1145/3313276.3316312
2. Bassily, R., Nissim, K., Smith, A.D., Steinke, T., Stemmer, U., Ullman, J.R.: Algorithmic stability for adaptive data analysis. SIAM J. Comput. **50**(3) (2021). https://doi.org/10.1137/16M1103646
3. Beimel, A., Nissim, K., Omri, E.: Distributed private data analysis: simultaneously solving how and what. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 451–468. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_25
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) CCS 1993, pp. 62–73. ACM (1993). https://doi.org/10.1145/168588.168596
5. Blocki, J., Blum, A., Datta, A., Sheffet, O.: The Johnson-Lindenstrauss transform itself preserves differential privacy. In: FOCS, pp. 410–419. IEEE Computer Society (2012). https://doi.org/10.1109/FOCS.2012.67
6. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data. IEEE Trans. Inf. Theory **44**(5), 1897–1905 (1998). https://doi.org/10.1109/18.705568
7. Brown, G., Bun, M., Feldman, V., Smith, A., Talwar, K.: When is memorization of irrelevant training data necessary for high-accuracy learning? In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 123–132 (2021). https://doi.org/10.1145/3406325.3451131
8. Bun, M., Nissim, K., Stemmer, U., Vadhan, S.P.: Differentially private release and learning of threshold functions. In: FOCS, pp. 634–649. IEEE Computer Society (2015). https://doi.org/10.1109/FOCS.2015.45
9. Bun, M., Ullman, J.R., Vadhan, S.P.: Fingerprinting codes and the price of approximate differential privacy. In: STOC, pp. 1–10. ACM (2014). https://doi.org/10.1145/2591796.2591877
10. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_33

11. Coretti, S., Dodis, Y., Guo, S.: Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 693–721. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_23
12. Coretti, S., Dodis, Y., Guo, S., Steinberger, J.: Random oracles and non-uniformity. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 227–258. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_9
13. Dai, W., Tessaro, S., Zhang, X.: Super-linear time-memory trade-offs for symmetric encryption. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12552, pp. 335–365. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64381-2_12
14. Dodis, Y., Farshim, P., Mazaheri, S., Tessaro, S.: Towards defeating backdoored random oracles: indifferentiability with bounded adaptivity. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12552, pp. 241–273. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64381-2_9
15. Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., Roth, A.L.: Preserving statistical validity in adaptive data analysis. In: Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, pp. 117–126 (2015)
16. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. J. Priv. Confidentiality **7**(3), 17–51 (2016). https://doi.org/10.29012/jpc.v7i3.405
17. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, pp. 381–390 (2009). https://doi.org/10.1145/1536414.1536467
18. Dwork, C., Rothblum, G.N., Vadhan, S.P.: Boosting and differential privacy. In: FOCS, pp. 51–60. IEEE Computer Society (2010). https://doi.org/10.1109/FOCS.2010.12
19. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_11
20. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS 2008, pp. 293–302. IEEE Computer Society (2008). https://doi.org/10.1109/FOCS.2008.56
21. Feldman, V.: Does learning require memorization? A short tale about a long tail. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, pp. 954–959 (2020). https://doi.org/10.1145/3357713.3384290
22. Göös, M., Lovett, S., Meka, R., Watson, T., Zuckerman, D.: Rectangles are nonnegative juntas. In: Servedio, R.A., Rubinfeld, R. (eds.) STOC 2015, pp. 257–266. ACM (2015). https://doi.org/10.1145/2746539.2746596
23. Göös, M., Pitassi, T., Watson, T.: Deterministic communication vs. partition number. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, pp. 1077–1088. IEEE (2015). https://doi.org/10.1109/FOCS.2015.70
24. Hardt, M., Ullman, J.R.: Preventing false discovery in interactive data analysis is hard. In: FOCS, pp. 454–463. IEEE Computer Society (2014). https://doi.org/10.1109/FOCS.2014.55
25. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28**(4), 1364–1396 (1999). https://doi.org/10.1137/S0097539793244708
26. Hazay, C., López-Alt, A., Wee, H., Wichs, D.: Leakage-resilient cryptography from minimal assumptions. J. Cryptol. **29**(3), 514–551 (2015). https://doi.org/10.1007/s00145-015-9200-x

27. Kalai, Y.T., Reyzin, L.: A survey of leakage-resilient cryptography. In: Goldreich, O. (ed.) Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, pp. 727–794. ACM (2019). https://doi.org/10.1145/3335741.3335768
28. Kothari, P.K., Meka, R., Raghavendra, P.: Approximating rectangles by juntas and weakly-exponential lower bounds for LP relaxations of CSPs. In: Hatami, H., McKenzie, P., King, V. (eds.) STOC 2017, pp. 590–603. ACM (2017). https://doi.org/10.1145/3055399.3055438
29. Mironov, I., Pandey, O., Reingold, O., Vadhan, S.P.: Computational differential privacy. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 126–142. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_8
30. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_27
31. Raz, R., McKenzie, P.: Separation of the monotone NC hierarchy. Comb. **19**(3), 403–435 (1999). https://doi.org/10.1007/s004930050062
32. Smith, A.D., Song, S., Thakurta, A.: The Flajolet-Martin sketch itself preserves differential privacy: private counting with minimal space. In: NeurIPS (2020). https://doi.org/10.5555/3495724.3497365
33. Standaert, F., Pereira, O., Yu, Y., Quisquater, J., Yung, M., Oswald, E.: Leakage resilient cryptography in practice. In: Sadeghi, A., Naccache, D. (eds.) Towards Hardware-Intrinsic Security - Foundations and Practice. Information Security and Cryptography, pp. 99–134. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14452-3_5
34. Steinke, T., Ullman, J.R.: Interactive fingerprinting codes and the hardness of preventing false discovery. In: COLT. JMLR Workshop and Conference Proceedings, vol. 40, pp. 1588–1628. JMLR.org (2015)
35. Tardos, G.: Optimal probabilistic fingerprint codes. J. ACM (JACM) **55**(2), 1–24 (2008). https://doi.org/10.1145/1346330.1346335
36. Ullman, J.: Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, pp. 361–370 (2013). https://doi.org/10.1145/2488608.2488653
37. Unruh, D.: Random oracles and auxiliary input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_12
38. Woodruff, D.P.: Optimal space lower bounds for all frequency moments. In: Munro, J.I. (ed.) Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, 11–14 January 2004, pp. 167–175. SIAM (2004). https://doi.org/10.5555/982792.982817