Black Box Crypto is Useless for Doubly Efficient PIR*

Wei-Kai Lin
University of Virginia

Ethan Mook
Northeastern University

Daniel Wichs†
Northeastern University
and NTT Research

March 25, 2025

Abstract

A *(single server) private information retrieval (PIR)* allows a client to read data from a public database held on a remote server, without revealing to the server which locations she is reading. In a *doubly efficient PIR (DEPIR)*, the database is first preprocessed offline into a data structure, which then allows the server to answer any client query efficiently in sub-linear online time. Constructing DEPIR is a notoriously difficult problem, and this difficulty even extends to a weaker notion secret-key DEPIR (SK-DEPIR), where the database is preprocessed using secret randomness and the client is given a secret key for making queries. We currently only have constructions of SK-DEPIR from the Ring LWE assumption or from non-standard code-based assumptions.

We show that the black-box use of essentially all generic cryptographic primitives (e.g., key agreement, oblivious transfer, indistinguishability obfuscation, etc.), including idealized primitives (e.g., random oracles, generic multilinear groups, virtual black-box obfuscation, etc.) is essentially useless for constructing SK-DEPIR. In particular, in any such SK-DEPIR construction, we can replace all black-box use of these primitives with just a black-box use of one-way functions. While we conjecture that SK-DEPIR cannot be constructed using black-box one-way functions alone, we are unable to show this in its full generality. However, we do show this for *2-round schemes with a passive server* that simply outputs requested locations in the preprocessed data structure, which is the format of all known schemes. Overall, this shows that the black-box use of essentially all crypto primitives is insufficient for constructing 2-round passive-server SK-DEPIR, and does not provide any benefit beyond black-box one-way functions for constructing general SK-DEPIR.

*©IACR 2025. This article is the final version submitted by the author(s) to the IACR and to Springer-Verlag on Feb 23, 2025. The version published by Springer-Verlag is available at DOI.

†Research supported by NSF grant CNS-1750795, CNS-2055510 and the Alfred P. Sloan Research Fellowship.

Contents

1 Introduction **1**
- 1.1 Our Results . . . 2
- 1.2 Our Techniques . . . 3

2 Preliminaries **5**
- 2.1 Definition of Secret-Key Doubly Efficient PIR . . . 5
- 2.2 Black-Box Constructions . . . 7

3 Crypto Oracles **7**
- 3.1 Constructing Obfuscation with a Crypto Oracle . . . 8
- 3.2 Instantiating the Generic Multilinear Group Model . . . 10

4 Removing Crypto Oracles from SK-DEPIR **11**
- 4.1 SK-DEPIR to Passive-Server SK-DEPIR . . . 13
- 4.2 Compiling Out the Crypto Oracle . . . 13
- 4.3 Discussion of Theorem 4.3 . . . 14

5 Ruling Out 2-Round Passive-Server SK-DEPIR **16**
- 5.1 Definition of Split-key SK-DEPIR . . . 17
- 5.2 Split the Key in SRF-DEPIR . . . 18
- 5.3 Ruling Out Split-key SK-DEPIR . . . 21

1 Introduction

PIR. A *(single-server) private information retrieval (PIR)* scheme [CGKS95, KO00] is a protocol between a server who holds a database $\mathsf{DB} \in \{0,1\}^N$ and a client who wishes to learn the i'th location $\mathsf{DB}[i]$ of the database without revealing the index $i \in [N]$ to the server. The main goal is to minimize the *communication complexity* between the server and the client to be sublinear in N and ideally just $\operatorname{poly}\log(N)$. Classical PIR comes with a major limitation in that it inherently requires the server to read the entire database DB during each protocol execution and therefore the *computational complexity* on the server side is at least linear in N; if the server did not access some subset of database locations, it would learn that the client's index does not fall within that set, which would violate privacy.

DEPIR. *Doubly efficient PIR (DEPIR)* [BIM00, CHR17, BIPW17, LMW23] gets around the above limitation and simultaneously achieves low communication and online server computation. It does so by allowing the database $\mathsf{DB} \in \{0,1\}^N$ to be *preprocessed* into a data structure $\widetilde{\mathsf{DB}}$ that is stored on the server. The one-time preprocessing is allowed to run in time that is linear, or even slightly super-linear, in the database size N. Subsequently, any client can run a PIR protocol with the server to learn the value $\mathsf{DB}[i]$ without revealing the index i, where both the communication and the server/client computation during the protocol must be sub-linear in N, and ideally just $\operatorname{poly}\log N$.

Flavors of DEPIR. One can define 3 flavors of DEPIR depending on the nature of the preprocessing: *unkeyed, public-key and secret-key DEPIR* in descending order of strength. Unkeyed DEPIR is the simplest and strongest notion, where the preprocessing that maps the database DB to the data structure $\widetilde{\mathsf{DB}}$ is a deterministic function that the server executes on its own, and subsequently any client can run a PIR protocol with the server without needing any keys. Public-key and secret-key DEPIR (PK-DEPIR, SK-DEPIR) both require a trusted party with secret random coins to setup the system by generating a key that is given to the client(s) and creating the preprocessed database $\widetilde{\mathsf{DB}}$ that is given on the server; in a public-key DEPIR the key can be public and known to the server, while in secret-key DEPIR it has to be secret and only known to the client(s) but not the server.

We say that a DEPIR scheme is a *passive server* DEPIR, if the server doesn't do any computation and the entire online protocol consists of the client looking up various locations i in the server's preprocessed data structure, and the server replying with the corresponding values $\widetilde{\mathsf{DB}}[i]$. In a *2-round passive-server* DEPIR, the client sends an entire set of locations I in the fist round and the server replies with $\widetilde{\mathsf{DB}}[I] = \{\widetilde{\mathsf{DB}}[i] \ : \ i \in I\}$ in the second round.

Prior work on DEPIR. Beimel, Ishai and Malkin [BIM00] proposed the notion of (unkeyed) DEPIR, and asked whether it is possible in the single server setting. Canetti, Holmgren and Richelson [CHR17] and Boyle, Ishai, Pass and Wootters [BIPW17] gave the first candidate constructions of keyed DEPIR. They showed how to construct SK-DEPIR under a new non-standard computational hardness assumption based on secretly permuted Reed-Muller codes. This assumption was further studied by [BHW19, BHMW21]. Boyle at al. [BIPW17] also showed how to upgrade SK-DEPIR to PK-DEPIR via a heuristic use of "ideal" (VBB) program obfuscation. The work of Lin, Mook and Wichs [LMW23] constructed the strongest notion of unkeyed DEPIR under the Ring

Learning with Errors (RingLWE) assumption. In that scheme, for any constant $\varepsilon > 0$, for database $\mathsf{DB} \in \{0,1\}^N$, the preprocessing run-time and the size of the preprocessed data structure $\widetilde{\mathsf{DB}}$ is $O(N^{1+\varepsilon})$, while the client and server run-times as well as the communication-complexity of each PIR query are just $\mathrm{poly}\log(N)$. All of these schemes are 2-round passive-server schemes.[1]

It remains a fascinating open problem to construct any flavor of DPEIR under any standard assumption beyond RingLWE, such as standard LWE, DDH (in bilinear maps), factoring, etc.

DEPIR within the Crypto Universe. DEPIR has a mysterious place within the crypto universe. We know that unkeyed DEPIR implies standard PIR which implies oblivious transfer [DMO00] and collision-resistant hashing [IKO05]. However, when it comes to PK-DEPIR or SK-DEPIR, we only know that they imply one-way functions [BIPW17], but do not know any other implications to other cryptographic primitives. Conversely, we do not know of *any* standard cryptographic primitives that would generically imply *any* flavor of DEPIR. This holds even if we consider powerful primitives, such as key agreement, oblivious transfer, fully homomorphic encryption, functional encryption, indistinguishability obfuscation, etc., and various combinations thereof. Moreover, it holds even if we consider powerful idealized primitives, such as random oracles, generic multilinear maps, VBB obfuscation, etc. This is in contrast to essentially all other cryptographic notions we can think of, which can be constructed generically from various combinations of sufficiently powerful primitives.

1.1 Our Results

In this work, we explain some of the mystery behind DEPIR and why it is so hard to construct from all other cryptographic primitives. Our main result shows that the black-box use of a large class of generic crypto primitives, including idealized ones and including all those listed above, is useless for constructing SK-DEPIR. In particular, the black-box use of such primitives can be replaced with just a black-box use of one-way functions while preserving efficiency and security.[2] We conjecture that there is no black-box construction of SK-DEPIR from just one-way functions. While we do not know how to prove this conjecture in its full generality, we do prove it for the special case of 2-round passive-server SK-DEPIR. Combining the results implies that there is no black-box construction of 2-round passive server DEPIR of any flavor (even SK-DEPIR) from a large class of generic crypto primitives. We elaborate on the results in more detail.

Idealized Cryptographic Primitives. We define a general notion of idealized cryptographic primitives that we call "crypto oracles". A crypto oracle $\mathcal{B}^R$ consists of a polynomial-time computable stateless and deterministic function $\mathcal{B}$ that can make oracle calls to a *secret random function* $R : \{0,1\}^* \to \{0,1\}$. When constructing cryptosystems with a crypto oracle $\mathcal{B}^R$, the schemes and the adversary can all make oracle calls to $\mathcal{B}^R$, but cannot call the function R directly. Various crypto oracles $\mathcal{B}^R$ allow us to easily construct essentially all cryptographic primitives we can think of, and we provide examples for multilinear maps, and VBB obfuscation for programs with oracle gates.[3]

[1] The work of [BIPW17] only focused on the restricted notion 2-round passive-server SK-DEPIR schemes by default, and referred to it as *oblivious locally decodable codes (OLDCs)*.

[2] As mentioned earlier, VBB obfuscation can be used to upgrade SK-DEPIR to PK-DEPIR, but our result shows that it cannot help construct SK-DEPIR itself.

[3] Note that the combination of several different crypto oracles $\mathcal{B}_1^{R_1}, \mathcal{B}_2^{R_2}, \ldots$ can be captured via one combined oracle $\mathcal{B}^R$ and therefore we can focus on the case of just a single oracle.

Removing Idealized Crypto. We convert any SK-DEPIR using some crypto oracle $\mathcal{B}^R$ into a SK-DEPIR where the client has black-box access to a secret random function R that is unknown to the server or the adversary. This transformation preserves the total client/server computational complexity. We can then replace R with a pseudorandom function to get a SK-DEPIR that only makes black-box use of one-way functions. Therefore, general crypto oracles $\mathcal{B}^R$ offer no help in constructing SK-DEPIR beyond what can be done with one-way functions. Moreover, this result shows that crypto oracles cannot help even in conjunction with other real-world assumptions – for example, any construction of SK-DEPIR from the hardness of factoring and a black-box use of VBB obfsucation can be converted into one based on the hardness of factoring alone. Under the conjecture that there is no black-box construction of SK-DEPIR from just one-way functions, this implies that there is no black-box construction of SK-DEPIR from any crypto oracle (without additional assumptions).

Impossibility for 2-Round Passive-Server SK-DEPIR. We show that there is no information-theoretically secure construction of 2-round passive-server SK-DEPIR in a model where the client has black-box access to a secret random function R. Together with our previous result, which is round preserving for passive-server protocols, this implies that there is no unconditionally secure construction of such schemes in the public random oracle model or a computationally secure one that only makes black-box use of one-way functions. Actually, this implies that there is no black-box construction of 2-round passive server SK-DEPIR (and therefore also PK-DEPIR or unkeyed DEPIR) from any crypto oracle.

Interpretation. The main interpretation of our result is that DEPIR requires concrete hardness assumptions about real-world problems (e.g., hardness of Ring LWE), which cannot be captured by a black-box use of crypto oracles. For example, we rule out a construction of 2-round passive-server SK-DEPIR in the generic group model. But we do not rule out such a construction from the discrete-logarithm assumption over a concrete group such as $\mathbb{Z}_p^*$, where the construction relies on the specific representation of the group elements.

1.2 Our Techniques

Our main technical idea is simple and is inspired by the elegant insight from a recent work Dujmovic and Hajiabadi [DH24]. They showed that standard PIR requires a linear number of black-box "public-key operations" for the server, where each such operation is a call to some ideal oracle.[4] Their main insight is that, for any PIR scheme where the server makes calls to some ideal oracle, we can convert it into a scheme in the plain model, by having the client play the role of the oracle and replacing the server's oracle calls with server/client communication; if the number of oracle calls in the original scheme is sub-linear then the communication would remain sub-linear. While the paper discussed applying this idea to DEPIR, it only showed that the *preprocessing phase* of DEPIR would require at least a linear number of "public-key operations". In particular, it did not offer any evidence that such DEPIR schemes cannot be constructed from ideal crypto primitives. We rely on their high-level idea as inspiration, but our result is technically different.

[4]The modeling of ideal cryptosystems in that work is more general than the crypto oracles of our work, and allows for stateful functionalities. However, we are not aware of any concrete interesting examples of crypto primitives where this is needed and that our notion fails to capture.

Active to Passive Server DEPIR. Firstly, we show how to convert any DEPIR into a passive-server DEPIR, where the server does not do any computation in the online phase, and the client simply asks for various locations i in the server's data structure and gets back the values $\widetilde{\mathsf{DB}}[i]$. The idea is to take an arbitrary DEPIR scheme and have the client simply run the server's computation locally; whenever the computation wants to access some data-structure location $\widetilde{\mathsf{DB}}[i]$, the client simply asks for it from the server and then continues the computation. Clearly the resulting passive-server DEPIR is at least as secure as the original one, and the total sum of server and client online computation also remains unchanged (although some work is shifted from the server to the client). The transformation may increase the number of rounds, since in each round of the original protocol may involve several sequential lookups in the data structure. This transformation applies to all flavors of (Unkeyed/PK/SK) DEPIR and also to DEPIR schemes that may rely on some crypto oracle.

Removing Crypto Oracles. Secondly, we can take any passive-server DEPIR scheme that relies on a crypto oracle $\mathcal{B}^R$ and convert it into a passive-server DEPIR scheme with a *secret random function* (SRF-DEPIR). In such a scheme, the preprocessing procedure and the client can make oracle calls to a secret random function $R : \{0,1\}^* \to \{0,1\}$, while security only holds against an adversarial server that does not get to make queries to R.

The idea is to simply take the original preprocessing procedure and client algorithms that make calls to $\mathcal{B}^R$ and have them perform the computation $\mathcal{B}$ themselves, while only making oracle calls to R (recalling that $\mathcal{B}$ is polynomial-time computable). Clearly, if originally security held even against servers making oracle calls to $\mathcal{B}^R$, then the security of the resulting SRF-DEPIR holds against servers without any oracle access. This transformation preserves the complexity and the number of rounds of the scheme. Note that if the original scheme is unconditionally secure in the crypto oracle model, then the resulting scheme will also be unconditionally secure with a secret random function. We can take any SRF-DEPIR scheme and convert it into a computationally secure SK-DEPIR with a black-box use of one-way functions, by replacing the random function R with a pseudorandom function whose key is the secret key. Combining the above two steps, we can convert any (unconditionally secure) SK-DEPIR that makes use of arbitrary crypto oracles into an (unconditionally secure) SRF-DEPIR scheme or an SK-DEPIR that only makes black-box use of a one-way functions.

Impossibility for 2-Round Passive-Server Schemes. Thirdly, we show that there does not exist any unconditionally secure 2-round passive-server SRF-DEPIR, where the preprocessing procedure and the client have oracle access to a secret random function $R : \{0,1\}^* \to \{0,1\}$. This also rules out 2-round passive-server SK-DEPIR in the random oracle model, which is stronger by also giving the adversary the power to query the random function. Therefore it also rules out 2-round passive-server SK-DEPIR from a black-box use of one-way functions.

Our proof proceeds in two steps. First, we convert the SRF-DEPIR (using R) into a special type of "split-key" SK-DEPIR in the plain model (without using R) where the secret key has several components that can be arbitrarily large in total, but there is only a small bounded polynomial-size *component of the secret key* that is simultanouesly used to preprocess the database and generate the client's PIR query. This small component essentially corresponds to the "heavy" inputs x in the SRF-DEPIR for which $R(x)$ has a sufficiently high likelihood of being called by the client to generate a PIR query. If we take sufficiently many of such heavy inputs, then it is unlikely that a

future PIR query will call $R(x')$ for any x' that is not in this component, but was involved in the preprocessing. Second, we show that such unconditionally secure "split-key"' SK-DEPIR cannot exist. We generalize a proof of [CHR17], which shows that the size of the secret key in any unconditionally secure SK-DEPIR has to grow with the number of PIR queries that the scheme supports. In particular, any such a scheme with a fixed polynomial size key cannot support an arbitrary polynomial number of PIR queries. We generalize this result to show that, even if the secret key can be arbitrarily large in total, the size of the secret key component that is simultanouesly used to preprocess the database and generate the client's PIR queries has to grow with the number of PIR queries supported. This yields a contradiction since, in the "split-key" SK-DEPIR we constructed, this component has bounded polynomial size, but the scheme supports an unbounded polynomial number of queries.

2 Preliminaries

Define $\mathbb{N} = \{0, 1, 2, \ldots\}$ to be the set of natural numbers, $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ to be the set of integers and $\mathbb{R}$ to be the set of real numbers. For any integer $n \geq 1$, define $[n] = \{1, \ldots, n\}$, and $[\![n]\!] = \{0, 1, \ldots, n-1\}$. For an array $A \in \{0,1\}^n$, we index the array from 0, and $A[i]$ denotes the bit in position $i \in [\![n]\!]$. By default, all our logarithms are base 2 and $\log n$ stands for $\log_2 n$. For any $q \in \mathbb{N}$, let $\mathbb{Z}_q$ be the ring $\mathbb{Z}/q\mathbb{Z}$. For a prime p, let $\mathbb{F}_p$ be the finite field of order p. A function $\nu : \mathbb{N} \to \mathbb{N}$ is said to be negligible, denoted $\nu(n) = \mathrm{negl}(n)$, if for every positive polynomial $p(\cdot)$ and all sufficiently large n it holds that $\nu(n) < 1/p(n)$. We use the abbreviation PPT for probabilistic polynomial time. For a finite set S, we write $a \leftarrow S$ to mean a is sampled uniformly randomly from S. For a randomized algorithm A, we let $a \leftarrow A(\cdot)$ denote the process of running $A(\cdot)$ and assigning the outcome to a; when A is deterministic, we write $a := A(\cdot)$ instead. We denote the security parameter by λ. For two distributions X, Y parameterized by λ we say that they are computationally indistinguishable, denoted by $X \approx_c Y$ if for every non-uniform PPT distinguisher D we have $|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| = \mathrm{negl}(\lambda)$; we say that they are statistically δ-close, if $\max_T(\Pr[X \in T] - \Pr[Y \in T]) \leq \delta$, denoted by $X \approx_\delta Y$.

2.1 Definition of Secret-Key Doubly Efficient PIR

A doubly-efficient private information retrieval (DEPIR) scheme is a protocol for a client to lookup from a database held by an adversarial server without revealing the location it's looking up. Rather than just store the database itself, the server stores some preprocessed version of the database to enable it to answer the client's query without having to read the whole database.

Several variants of DEPIR have been considered in the literature. They are called secret-key, public-key, and unkeyed DEPIR depending on the how and by whom the key is generated [BIM00, CHR17, BIPW17, LMW23]. We focus on the weakest variant, secret-key DEPIR, to make our barrier stronger. For the same reason, we relax the definition in the following ways: we allow imperfect correctness; we allow the online protocol many rounds of interaction rather than just two; we allow the client and the server to have arbitrary (polynomial) run time in the online protocol, and we only enforce that the number of locations read by the server is small (i.e. sublinear).

Definition 2.1 (Secret-key DEPIR)**.** *Let λ be the security parameter and $N = \mathrm{poly}(\lambda)$. A* secret-key doubly efficient private information retrieval (SK-DEPIR) *consists of the following algorithms and protocol.*

- $k \leftarrow \mathsf{KeyGen}(1^\lambda)$ *takes the security parameter* 1^λ*, and it randomly samples a key* k*.*
- $\widetilde{\mathsf{DB}} \leftarrow \mathsf{Prep}(1^\lambda, k, \mathsf{DB})$ *takes the security parameter* 1^λ*, a key* k *and a database* $\mathsf{DB} \in \{0,1\}^N$*, and outputs a preprocessed database* $\widetilde{\mathsf{DB}}$*.*
- *A two-party protocol* $\Pi(\widetilde{\mathsf{DB}}\ ;\ (N,k,i))$ *between the server who initially holds the preprocessed database* $\widetilde{\mathsf{DB}}$ *and the client who initially holds the database size* N*, a key* k *and index* $i \in [\![N]\!]$*. At the end of the protocol* Π*, the client outputs a bit* b *and the server outputs nothing. When needed, we explicitly denote* Π *as a pair of interactive algorithms* $(\mathsf{Query}(N,k,i), \mathsf{Resp}(\widetilde{\mathsf{DB}}))$*, where* Query *outputs the bit* b *at the end.*

The algorithms and and protocol should satisfy the following correctness, efficiency, and security properties.

ε-Correctness. *Honest execution of DEPIR successfully recovers requested data items except with* $\varepsilon(\lambda)$ *probability. That is, for every* $\mathsf{DB} \in \{0,1\}^N$ *and every* $i \in [\![N]\!]$*, it holds that*

$$\Pr\left[\Pi(\widetilde{\mathsf{DB}}; (N,k,i)) = \mathsf{DB}[i] \ :\ \begin{array}{rl} k & \leftarrow \mathsf{KeyGen}(1^\lambda); \\ \widetilde{\mathsf{DB}} & \leftarrow \mathsf{Prep}(1^\lambda, k, \mathsf{DB}) \end{array}\right] \geq 1 - \varepsilon(\lambda).$$

We say simply that a SK-DEPIR is correct *if* $\varepsilon(\lambda) = \mathrm{negl}(\lambda)$*.*

Efficiency. *We require that all of the algorithms of the scheme run in polynomial time in their inputs. We model* Resp *as having random access to the preprocessed database* $\widetilde{\mathsf{DB}}$*, and we say the* locality *of a SK-DEPIR is the maximum number* $\ell(\lambda, N)$ *of distinct locations* Resp *touches in* $\widetilde{\mathsf{DB}}$ *in the worst case. We require that the locality of the scheme is sublinear in* N*.*[5]

δ-Security. *No efficient adversary can distinguish the protocol* Π *that computes on input index* i_0 *or* i_1 *with better than* $\delta(\lambda)$ *advantage. Formally, define the following security game between a challenger and a semi-honest and stateful adversary* $\mathcal{A}$ *for any database* $\mathsf{DB} \in \{0,1\}^N$ *and any sequence of indices* $i_0, i_1, j_1, j_2, j_3, \cdots \in [\![N]\!]$*:*

1. *The challenger selects a random bit* $b \leftarrow \{0,1\}$*, generates a key* $k \leftarrow \mathsf{KeyGen}(1^\lambda)$*, and then preprocesses the database* $\widetilde{\mathsf{DB}} \leftarrow \mathsf{Prep}(1^\lambda, k, \mathsf{DB})$*. It sends* $\widetilde{\mathsf{DB}}$ *to* $\mathcal{A}$*.*
2. *Repeat the following for* $t = 1, 2, 3, \ldots$ *an arbitrary polynomial number of times until the adversary* $\mathcal{A}$ *proceeds with the next step:*
   (a) *$\mathcal{A}$ and the challenger (honestly) run the protocol* $\Pi(\widetilde{\mathsf{DB}}; (N,k,j_t))$*.*[6]
3. *(Challenge.) The adversary* $\mathcal{A}$ *and the challenger run the protocol* $\Pi(\widetilde{\mathsf{DB}}; (N,k,i_b))$*.*
4. $\mathcal{A}$ *outputs a guess bit* b'*.*

[5] For the case of positive results, one would ideally want Prep to run in nearly linear time in N and Π to run in polylogarithmic time in N (which implies polylogarithmic locality), and this is what [LMW23] achieve. The notion we define here is weaker, allowing any $o(N)$ locality and arbitrary polynomial run time, which strengthens our negative results.

[6] Notice that each invocation of Π inherits nothing from a previous invocation, but the stateful $\mathcal{A}$ can keep the transcripts throughout the game.

We require that for every non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that the distinguishing advantage of $\mathcal{A}$ in the above security game is $|\Pr[b' = b] - 1/2| = \delta(\lambda)$. We say simply the scheme is secure *if $\delta(\lambda) = \mathrm{negl}(\lambda)$.*

Throughout this work we will also consider SK-DEPIR that exists in an idealized world relative to an oracle. We define such an SK-DEPIR as follows.

Definition 2.2. *For an arbitrary distribution $\mathcal{O}$ over oracles O, a SK-DEPIR constructed relative to $\mathcal{O}$ is a tuple $(\mathsf{KeyGen}^O, \mathsf{Prep}^O, \Pi^O)$ of O-oracle aided algorithms with syntax, efficiency and correctness as in Definition 2.1, where correctness also holds over the sampling of $O \leftarrow \mathcal{O}$. Such a scheme is secure if for all O-oracle aided adversaries $\mathcal{A}$ that make at most $q = \mathrm{poly}(\lambda)$ many queries to O, $\mathcal{A}$ has negligible advantage in the security game of Definition 2.1.*

2.2 Black-Box Constructions

Definition 2.3 (Black-box constructions [RTV04])**.** *A black-box construction of a primitive $\mathcal{Q}$ from a primitive $\mathcal{P}$ consists of two PPT algorithms (Q, S) as follows:*[7]

1. *Implementation: For any oracle P that implements $\mathcal{P}$, Q^P implements $\mathcal{Q}$.*
2. *Security reduction: for any oracle P implementing $\mathcal{P}$ and for any (computationally unbounded) oracle adversary A successfully breaking the security of Q^P, it holds that $S^{P,A}$ breaks the security of P.*

Definition 2.4 (Primitives in relative to an oracle [RTV04])**.** *A primitive $\mathcal{P}$ exists (unconditionally) relative to an idealized oracle $\mathcal{I}$ if there exists an implementation P of $\mathcal{P}$ which is computable by a PPT oracle machine with access to $\mathcal{I}$ and such that no adversary that makes polynomially many queries to $\mathcal{I}$ but has otherwise unbounded run time breaks P.*

Lemma 2.5 (Relativizing reduction [RTV04, MMN+16])**.** *If there is a black-box construction of $\mathcal{Q}$ from $\mathcal{P}$ then:*

1. *If $\mathcal{P}$ exists relative to idealized model $\mathcal{I}$, $\mathcal{Q}$ exists relative to $\mathcal{I}$ as well.*
2. *If in addition the construction of $\mathcal{P}$ from $\mathcal{I}$ is black-box, then a black-box construction of $\mathcal{Q}$ relative to $\mathcal{I}$ exists as well.*

3 Crypto Oracles

In this section we introduce the notion of a *crypto oracle* that captures a large and powerful class of oracles.

Definition 3.1. *A* crypto oracle *is a function $\mathcal{B}^R$ where $\mathcal{B}$ is a stateless, polynomial time, deterministic Turing machine with oracle access to a secret random function $R : \{0,1\}^* \to \{0,1\}$.*

[7] The definition we give here corresponds to that of *fully black-box* constructions in the literature. We focus on this definition for the sake of conceptual clarity. However, we note that in many cases, our separations here actually extend to semi black-box constructions (also defined in [RTV04]). See Remarks 4.10 and 5.3 for further discussion.

Formally $\mathcal{B}$ is implemented as a single Turing machine but we often will think of $\mathcal{B}$ as having multiple component algorithms that can each be queried separately. In such cases we write $\mathcal{B}^R$ as a tuple $\mathcal{B}^R = (\mathcal{B}_1^R, \ldots, \mathcal{B}_k^R)$. Also note that given a single secret random function R, one can derive arbitrarily many independent random functions $R_1, \ldots, R_k : \{0,1\}^* \to \{0,1\}$ from R by setting $R_i(\cdot) = R(\mathsf{pre}_i, \cdot)$ for some arbitrary set of equal length, distinct prefixes $\mathsf{pre}_1, \ldots, \mathsf{pre}_k$. Given this, observe that any sequence $\mathcal{B}_1^{R_1}, \ldots \mathcal{B}_k^{R_k}$ of independent crypto oracles can be combined into a single crypto oracle $\mathcal{B}^R = (\mathcal{B}_1^{R_1}, \ldots, \mathcal{B}_k^{R_k})$ where each of the functions R_i are independently derived from R.

In the remainder of this section, we demonstrate that crypto oracles capture a large array of cryptographic functionalities. One can immediately see that the random oracle model can be viewed as a crypto oracle (by setting $\mathcal{B}$ to the Turing machine that transparently forwards queries to its oracle). However by leveraging the secrecy of the function R, many more advanced cryptographic primitives can be implemented by a crypto oracle. We show two powerful examples below. First we show how to implement VBB obfuscation for Turing machines as a crypto oracle, and later we show how to instantiate a generic multilinear group.

3.1 Constructing Obfuscation with a Crypto Oracle

Theorem 3.2. *There exists a crypto oracle $\mathcal{B}^R = (\mathsf{Obf}^R, \mathsf{Eval}^R)$ that implements VBB obfuscation for Turing machines. More formally, $\mathcal{B}^R$ satisfies:*

- ***Correctness:*** *With probability 1 over the sampling of R, for any Turing machine M, input x and randomness r, $\mathsf{Eval}^R(\mathsf{Obf}^R(M, r), x, 1^T) = z$ where z is the result of running $M(x)$ for T steps.*
- ***Security:*** *There exists a simulator Sim such that for every oracle-aided algorithm $\mathcal{A}$ that makes at most $q = \mathrm{poly}(\lambda)$ many queries to $\mathcal{B}^R$,[8] and any Turing machine M,*

$$\left| \Pr_r \left[\mathcal{A}^{\mathcal{B}^R}(\mathsf{Obf}^R(M, r)) = 1 \right] - \Pr \left[\mathsf{Sim}^{\mathcal{A}, \mathcal{B}^R, M}(1^\lambda, 1^{|M|}) = 1 \right] \right| = \mathrm{negl}(\lambda). \tag{1}$$

Moreover, the above also holds in the presence of a separate, independent crypto oracle $\overline{\mathcal{B}}^{\overline{R}}$, and can achieve VBB obfuscation for $\overline{\mathcal{B}}^{\overline{R}}$-oracle aided Turing machines. More specifically, for any crypto oracle $\overline{\mathcal{B}}^{\overline{R}}$, there exists a combined crypto oracle $\mathcal{B}^{R,\overline{R}} = (\mathsf{Obf}^{R,\overline{R}}, \mathsf{Eval}^{R,\overline{R}}, \overline{\mathcal{B}}^{\overline{R}})$ that both implements functionality of $\overline{\mathcal{B}}^{\overline{R}}$ and satisfies the correctness and security properties as above with the modification that the Turing machine M now has oracle access to $\overline{\mathcal{B}}^{\overline{R}}$.

Before we proceed to the proof of the above theorem, we first prove the following lemma, which shows that we can use a crypto oracle to construct a (slightly strengthened) form of authenticated encryption, that we will use in the proof of the theorem. The construction largely follows the standard template, using the secret random function R in place of a PRF.

Lemma 3.3. *There exists a crypto oracle $\mathcal{B}^R = (\mathsf{Enc}^R, \mathsf{Dec}^R)$ that implements perfectly correct authenticated encryption. That is, $\mathcal{B}^R$ satisfies the following properties:*

- ***Correctness:*** *With probability 1 over the sampling of R, for all $\lambda \in \mathbb{N}$, messages $x \in \{0,1\}^\lambda$, and randomness $r \in \{0,1\}^\lambda$, $\mathsf{Dec}^R(\mathsf{Enc}^R(x, r)) = x$.*

[8] Aside from the bound on the number of oracle queries it makes, the run time of $\mathcal{A}$ is otherwise unbounded.

- ***Hiding with chosen ciphertext and randomness:*** *For any oracle-aided two-part adversary* $(\mathcal{A}_1, \mathcal{A}_2)$ *that makes at most* $q = \mathrm{poly}(\lambda)$ *queries to* $\mathcal{B}^R$*, we have*

$$\left|\Pr\left[\mathcal{A}_2^{\mathcal{B}^R_{y^* \to \perp}}(y^*, \mathsf{st}) = 1\right] - \Pr\left[\mathcal{A}_2^{\mathcal{B}^R_{u \to \perp}}(u, \mathsf{st}) = 1\right]\right| = \mathrm{negl}(\lambda), \tag{2}$$

  where, in the above, $\mathcal{B}^R_{c \to \perp}$ *denotes the oracle* $\mathcal{B}^R$ *with the modification that we set* $\mathsf{Dec}^R(c) = \perp$ *and the probability space is defined over sampling* $(x^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{B}^R}(1^\lambda)$*, setting* $y^* = \mathsf{Enc}^R(x^*, r^*)$ *for uniformly random* $r^* \leftarrow \{0,1\}^\lambda$*, sampling* $u \leftarrow \{0,1\}^{|y^*|}$*, as well as the random coins of* $\mathcal{A}_2$*.*

Proof. The oracle $\mathcal{B}^R = (\mathsf{Enc}^R, \mathsf{Dec}^R)$ is defined as follows. Let $R_1, R_2, R_3 : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be independent random functions derived from R.

$\mathsf{Enc}^R(x, r)$**:** Compute $y_1 = R_1(r)$, $y_2 = R_2(y_1) \oplus x$ and $y_3 = R_3(y_1 \| y_2)$. Output $y = (y_1, y_2, y_3)$.

$\mathsf{Dec}^R(y)$**:** Parse $y = (y_1, y_2, y_3)$. Check that $y_3 = R_3(y_1 \| y_2)$, if it is not output $\perp$. Otherwise, output $R_2(y_1) \oplus y_2$.

Correctness follows immediately by inspection. To argue security we define the following sequence of hybrid experiments:

- $\mathbf{H}_0$: This is the distribution on all of the random variables in the experiment on the left hand side of equation 2.
- $\mathbf{H}_1$: We change how the challenge ciphertext $y^* = (y_1, y_2, y_3)$ is sampled. We sample y_1 uniformly randomly and independently instead of setting $y_1 = R_1(r^*)$.
- $\mathbf{H}_2$: We again change how y^* is sampled. Now we sample y_2 uniformly randomly and independently as well instead of setting $y_2 = R_2(y_1) \oplus x^*$.
- $\mathbf{H}_3$: This is the distribution on all of the random variables in experiment on the right hand side of equation 2 where the challenge ciphertext is fully uniform.

Conditioned on the event that R_1 is never queried on r^*, $\mathbf{H}_0$ is identically distributed to $\mathbf{H}_1$. Thus the statistical distance between $\mathbf{H}_0$ and $\mathbf{H}_1$ is bounded by the probability that R_1 is queried on r^*. But since R_1 is a random function, r^* is independent from the remainder of the experiment. Then since each query the adversary makes results in only one query to R_1, at most $q + 1$ queries are made to R_1 throughout the experiment. Thus the probability that R_1 is queried on r^* is at most $(q+1)/2^\lambda = \mathrm{negl}(\lambda)$.

Now, similarly, conditioned on the event that R_2 is never queried on y_1, $\mathbf{H}_1$ is identically distributed to $\mathbf{H}_2$. And conditioned on the event that R_3 is never queried on $y_1 \| y_2$, $\mathbf{H}_2$ is identical to $\mathbf{H}_3$. Thus the same argument as above yields that the statistical distance between $\mathbf{H}_0$ and $\mathbf{H}_3$ is negligible. This completes the proof of Lemma 3.3. □

Now we are ready to prove the theorem.

Proof of Theorem 3.2. We define the oracle $\mathcal{B}^R = (\mathsf{Obf}^R, \mathsf{Eval}^R)$ as follows. We use R to instantiate the encryption scheme from Lemma 3.3.

$\mathsf{Obf}^R(M, r)$**:** Output $\mathsf{Enc}^R(M, r)$.

$\mathsf{Eval}^R(\tilde{M}, x)$**:** Let $M' = \mathsf{Dec}^R(\tilde{M})$. If $M' = \perp$, output $\perp$. Otherwise, output $M'(x)$.

Correctness follows immediately by inspection. Next we argue security. Define the simulator Sim as follows:

$\mathsf{Sim}^{\mathcal{A},\mathcal{B}^R,M}(1^\lambda, 1^{|M|})$**:** Sample a uniformly random dummy ciphertext $u \leftarrow \{0,1\}^\ell$ where $\ell = |M| + 2\lambda$ is the length of an obfuscation of M, then run $\mathcal{A}(u)$.

Whenever $\mathcal{A}$ makes a query to $\mathsf{Eval}^R(u, x)$, query the oracle M for $M(x)$ and forward it to $\mathcal{A}$. Whenever $\mathcal{A}$ makes any other query to its oracle, simply forward the query to the $\mathcal{B}^R$ and send its response back to $\mathcal{A}$.

When $\mathcal{A}$ halts, halt and output as it does.

It is easy to see that VBB security then follows from the security of the encryption scheme. Let $\mathcal{A}$ be an adversary against VBB security for some Turing machine M. We define an adversary $\overline{\mathcal{A}} = (\overline{\mathcal{A}}_1, \overline{\mathcal{A}}_2)$ for the encryption scheme as follows: $\overline{\mathcal{A}}_1$ simply outputs M. Then, on input y which is either an encryption of M or a random string, $\overline{\mathcal{A}}_2$ runs $\mathcal{A}(y)$. Whenever $\mathcal{A}$ queries for evaluation on (y, x), reply with $M(x)$; whenever $\mathcal{A}$ makes any other use of its oracles $(\mathsf{Obf}^R, \mathsf{Eval}^R)$, use $(\mathsf{Enc}^R, \mathsf{Dec}^R)$ to emulate their responses following the definition of $(\mathsf{Obf}^R, \mathsf{Eval}^R)$ above.

In the case that y is an encryption of M, $\overline{\mathcal{A}}$ perfectly emulates the experiment on the left hand side of Equation (1), and, in the case that y was a random string, $\overline{\mathcal{A}}$ perfectly emulates the experiment on the right hand side. Thus by security of the encryption scheme, $\overline{\mathcal{A}}$ must have negligible advantage, and hence $\mathcal{A}$ must as well.

To achieve VBB obfuscation for Turing machines aided by some independent crypto oracle $\overline{\mathcal{B}}^{\overline{R}}$ the above construction needs only minor modification. Specifically, in the Eval^R, when evaluating the decrypted Turing machine M, simply use an independent random function $\overline{R}$ to emulate the responses to any queries M makes to the other crypto oracle $\overline{\mathcal{B}}^{\overline{R}}$. The proof of security is unaffected. □

3.2 Instantiating the Generic Multilinear Group Model

Definition 3.4. *The* generic multilinear group *is an ensemble of distributions $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ parameterized by $\tau \in \mathbb{N}$, $p \geq 2^\lambda$ and $\ell \geq \log p$ where each $\mathcal{M}_\lambda$ is the distribution of oracles defined by sampling functions $(\{L_i\}_{i\in[\tau+1]}, \{\mathsf{add}_i\}_{i\in[\tau+1]}, e)$ as follows.*

- *The functions $L_1, \ldots, L_\tau, L_{\tau+1} : \mathbb{Z}_p \to \{0,1\}^\ell$ are sampled as independent, uniformly random injections.*
- *For each $i \in [\tau+1]$, the function $\mathsf{add}_i(\{0,1\}^\ell)^2 \times \mathbb{Z}_p^2 \to \{0,1\}^\ell \cup \perp$ is defined by setting*

$$\mathsf{add}_i(y_0, y_1, \alpha_0, \alpha_1) = L_i(\alpha_0 x_0 + \alpha_1 x_1)$$

  if there exist $x_0, x_1 \in \mathbb{Z}_p$ such that $L_i(x_0) = y_0$ and $L_i(x_1) = y_1$, and $\mathsf{add}(y_0, y_1, \alpha_0, \alpha_1) = \perp$ otherwise.

- *The function $e : (\{0,1\}^\ell)^\tau \to \{0,1\}^\ell \cup \perp$ is defined by setting*

$$e(y_1, \ldots, y_\tau) = L_{\tau+1}(x_1 x_2 \cdots x_\tau)$$

  if, for all $i \in [\tau]$, there exists x_i such that $L_i(x_i) = y_i$, and setting $e(y_1, \ldots, y_\tau) = \perp$ otherwise.

Theorem 3.5. *For any $\tau \in \mathbb{N}$, $p = p(\lambda)$ and $\ell = \ell(\lambda)$, let $\mathcal{M}$ be the generic multilinear group $\mathcal{M}$ with parameters τ, p, ℓ. There exists an ensemble of crypto oracles $\mathcal{B}_\lambda^R$, such that any oracle-aided adversary $\mathcal{A}$ that makes at most $q = \mathrm{poly}(\lambda)$ oracle queries cannot distinguish $\mathcal{M}$ from $\mathcal{B}^R$:*

$$\left| \Pr_{M \leftarrow \mathcal{M}_\lambda} \left[\mathcal{A}^M(1^\lambda) = 1 \right] - \Pr_R \left[\mathcal{A}^{\mathcal{B}_\lambda^R}(1^\lambda) = 1 \right] \right| = \mathrm{negl}(\lambda).$$

Before we prove the above theorem, we first recall the following classic result that iterating Feistel networks over a random function yields a permutation that is indistinguishable from random given only polynomially many queries. This implies the existence of a pseudorandom permutation relative to a crypto oracle.

Lemma 3.6 (Adapted from [LR86])**.** *There exists a crypto oracle $\mathcal{B}^R = P^R$ that implements a pseudorandom permutation secure against all unbounded adversaries that make at most $q = \mathrm{poly}(\lambda)$ queries to P^R. Moreover, given oracle access to R, P^R is efficiently invertible.*

Now we are ready to prove the theorem.

Proof of Theorem 3.5. We construct the crypto oracle

$$\mathcal{B}_\lambda^R = \left(\left\{ L_i^R \right\}_{i \in [\tau+1]}, \left\{ \mathsf{add}_i^R \right\}_{i \in [\tau+1]}, e^R \right)$$

as follows. We first implement $(\tau+1)$-many PRPs $P_1^{R_1}, \ldots, P_{\tau+1}^{R_{\tau+1}} : \{0,1\}^\ell \to \{0,1\}^\ell$ via Lemma 3.6, where $R_1, \ldots, R_{\tau+1}$ are independent random functions derived from R. Since $\ell \geq \log p$, we can interpret $\mathbb{Z}_p$ as a subset of $\{0,1\}^\ell$ in the standard way.

- For each $i \in [\tau+1]$, define $L_i^R(x) = P_i^{R_i}(x)$.
- For each $i \in [\tau+1]$, define $\mathsf{add}_i(y_0, y_1, \alpha_0, \alpha_1)$ to first compute $x_b = (P_i^{R_i})^{-1}(y_b)$ for $b \in \{0,1\}$ and check that $x_0, x_1 \in \mathbb{Z}_p$. If so, output $P_i^{R_i}(\alpha_0 x_0 + \alpha_1 x_1)$, otherwise output $\perp$.
- Define $e(y_1, \ldots, y_\tau)$ to first compute $x_i = (P_i^{R_i})^{-1}(y_i)$ and check that $x_i \in \mathbb{Z}_p$ for all $i \in [\tau]$. If so, output $P_{\tau+1}^{R_{\tau+1}}(y_1 \cdots y_\tau)$, otherwise output $\perp$.

It is immediate to see that $\mathcal{B}_\lambda^R$ is indistinguishable from M by a standard reduction to the security of the pseudorandom permutations. □

4 Removing Crypto Oracles from SK-DEPIR

In this section we prove our main technical result showing that any SK-DEPIR scheme that is constructed relative to a crypto oracle can be compiled into one that only relies on the client having access to a secret random function.

Definition 4.1 (SK-DEPIR in the Secret Random Function Model)**.** *A* SK-DEPIR in the secret random function model (SRF-DEPIR) *is a SK-DEPIR* $(\mathsf{KeyGen}^R, \mathsf{Prep}^R, \Pi = (\mathsf{Query}^R, \mathsf{Resp}))$ *constructed relative to a random function R (see Definition 2.2) with the modifications that the honest server algorithm* Resp *does not have access to R and, for the purpose of security, we quantify over adversaries that do not have access to the oracle R (but may have unbounded run time).*

As a stepping stone toward our main result, we first convert to a *passive server* SK-DEPIR. Informally, this is a SK-DEPIR where in the online protocol Π the only communication is the client sending desired read locations and the server responding with the values at those locations from the preprocessed database.

Definition 4.2 (Passive-server SK-DEPIR)**.** *A secret-key DEPIR* $(\mathsf{KeyGen}, \mathsf{Prep}, \Pi)$ *(Definition 2.1) is* passive-server *if the* Resp *algorithm of the protocol* $\Pi = (\mathsf{Query}, \mathsf{Resp})$ *is passive. Elaborately,* Resp *is the following interactive algorithm:*

1. $\mathsf{Resp}(\widetilde{\mathsf{DB}})$*: On input a preprocessed database* $\widetilde{\mathsf{DB}} \in \{0,1\}^{\tilde{N}}$*, where* $\tilde{N}$ *is the output length of the preprocessing. Then, upon receiving a message of the form* $I \subseteq [\![\tilde{N}]\!]$*, it responds with the assignment* $\widetilde{\mathsf{DB}}[I] := \{(i, \widetilde{\mathsf{DB}}[i])\}_{i \in I}$.

We also analogously define the notions of passive-server SK-DEPIR constructed relative to an oracle and passive-server SK-DEPIR in the secret random function model. In those cases we define security as in Definitions 2.2 and 4.1 respectively.[9]

For convenience, in the passive-server context, we often omit the server algorithm Resp *and simply write the tuple* $(\mathsf{KeyGen}, \mathsf{Prep}, \mathsf{Query})$ *to denote such a SK-DEPIR.*

We are now ready to state the main technical theorem of this section.

Theorem 4.3. *Suppose there is a (possibly active-server) SK-DEPIR* $(\mathsf{KeyGen}^{\mathcal{B}^R}, \mathsf{Prep}^{\mathcal{B}^R}, \Pi^{\mathcal{B}^R})$ *that exists relative to a crypto oracle* $\mathcal{B}^R$*. Then that scheme can be compiled into a passive-server SRF-DEPIR* $(\overline{\mathsf{KeyGen}}^R, \overline{\mathsf{Prep}}^R, \overline{\mathsf{Query}}^R)$.

This compilation preserves the correctness, security and locality original scheme, and the compiled scheme requires at most 2ℓ *rounds of communication where* ℓ *is the locality of the original scheme. Moreover if the original scheme was already passive-server, then the compiled scheme also preserves its round complexity.*

The above theorem follows immediately from Lemmas 4.4 and 4.5 that we state and prove in Sections 4.1 and 4.2 respectively. At a high level, our compilation proceeds in two steps:

1. First we convert the SK-DEPIR into a passive-server one. To do so the client will simply emulate the server's online algorithm Resp "in it's head" and ask the server any time Resp would read from the preprocessed database. This transformation preserves run times and locality but possibly blows up the round complexity of the protocol to 2ℓ where ℓ is the locality of the original scheme. Also this transformation is agnostic to the presence of any oracle the original scheme was constructed relative to, that is, the new passive-server scheme will be relative to the same crypto-oracle as the original scheme.

2. Second we convert from needing a crypto oracle to a scheme in the secret random function model. To do this, we simply observe that the now passive server never needs to query the crypto oracle, so we can just have the client emulate the crypto oracle $\mathcal{B}$ for itself, with the help of the secret random function. This transformation preserves locality and round complexity.

Following the proof of Theorem 4.3 via the lemmas in Sections 4.1 and 4.2, in Section 4.3 we discuss how to interpret this result. Most notably, we note that the secret random function model can be instantiated assuming the existence of one-way functions.

[9] We note that in the passive-server setting, the SRF model requirement that the honest server doesn't get access to the oracle R is automatically satisfied.

4.1 SK-DEPIR to Passive-Server SK-DEPIR

In this section we give the first step in our compilation, compiling any SK-DEPIR to a passive-server one. Our construction is as follows.

Construction. Consider a SK-DEPIR $(\mathsf{KeyGen}, \mathsf{Prep}, \Pi = (\mathsf{Query}, \mathsf{Resp}))$ that is possibly constructed relative to an oracle. To convert this SK-DEPIR into a passive-server SK-DEPIR, the intuition is simple: the client simply emulates the original server computation, and queries the passive server to access $\widetilde{\mathsf{DB}}$.[10] We formally construct the interactive algorithm $\mathsf{psQuery}$ as follows.

1. $\mathsf{psQuery}(N, k, i)$: on input database size N, key k and index $i \in [\![N]\!]$, run the interactive algorithm $\mathsf{Query}(N, k, i)$ that results in a message q and a state $\mathsf{st}_{\mathsf{Query}}$.
2. Repeat the following until Query halts:
   (a) Emulate the interactive algorithm Resp on the message q: when Resp needs to read $\widetilde{\mathsf{DB}}$ on indices $I = (i_1, i_2, \ldots, i_m)$ for some m, forward I to the passive server and obtain $\widetilde{\mathsf{DB}}[I]$ to fulfill Resp. Moreover, when Resp needs randomness, sample and supply uniformly random coins. Let $(\mathsf{ans}, \mathsf{st}_{\mathsf{Resp}})$ be the resulting message and state.
   (b) Perform Query on the message ans and state $\mathsf{st}_{\mathsf{Query}}$.
   (c) If the above Query halts with an output bit, then $\mathsf{psQuery}$ halts and outputs the same bit. Otherwise, the protocol continues as Query sends a message q' and keeps a state $\mathsf{st}'_{\mathsf{Query}}$, and thus $\mathsf{psQuery}$ continues by setting $(q, \mathsf{st}_{\mathsf{Query}}) \leftarrow (q', \mathsf{st}'_{\mathsf{Query}})$.

We claim that $(\mathsf{KeyGen}, \mathsf{Prep}, \mathsf{psQuery})$ is a passive-server SK-DEPIR.

Lemma 4.4. *For any SK-DEPIR $(\mathsf{KeyGen}, \mathsf{Prep}, \Pi = (\mathsf{Query}, \mathsf{Resp}))$ (possibly constructed relative to an oracle), the scheme $(\mathsf{KeyGen}, \mathsf{Prep}, \mathsf{psQuery})$ is a passive-server DEPIR (relative to the same oracle), where $\mathsf{psQuery}$ is constructed using $(\mathsf{Query}, \mathsf{Resp})$ as above. The compilation preserves correctness and security. If ℓ is the locality of the original scheme, then the passive-server scheme also has locality ℓ and requires at most 2ℓ rounds of interaction.*

Proof. The correctness, locality and passivity of the server follow directly from the construction.

The security follows by a standard reduction, sketched next. Assume for contradiction, an adversary $\mathcal{A}$ breaks the compiled scheme in the security game (Definition 2.1). Then, we construct another adversary $\mathcal{A}'$ that breaks the original scheme as follows. In the game, $\mathcal{A}'$ emulates the active server, and $\mathcal{A}'$ forwards to $\mathcal{A}$ its $\widetilde{\mathsf{DB}}$ and all its accesses to $\widetilde{\mathsf{DB}}$ during the emulation; finally, $\mathcal{A}'$ outputs the guess bit of $\mathcal{A}$. Recall that the security game considers only semi-honest adversaries. Hence, $\mathcal{A}'$ perfectly emulates the challenger of $\mathcal{A}$, and thus $\mathcal{A}'$ breaks the original scheme, a contradiction. □

4.2 Compiling Out the Crypto Oracle

In this section we give the second step in our compilation, removing the crypto oracle to convert the scheme into one in the secret random function model.

[10] A similar conversion is also used in a previous construction [DMZ23].

Lemma 4.5. *Suppose there is a passive server SK-DEPIR* $(\mathsf{KeyGen}^{\mathcal{B}^R}, \mathsf{Prep}^{\mathcal{B}^R}, \mathsf{Query}^{\mathcal{B}^R})$ *that exists relative to a crypto oracle* $\mathcal{B}^R$*. Then that scheme can be compiled into a passive server SRF-DEPIR* $(\overline{\mathsf{KeyGen}}^R, \overline{\mathsf{Prep}}^R, \overline{\mathsf{Query}}^R)$.

This compilation preserves the correctness, security, round complexity, and locality of the original scheme.

Proof. We construct the passive-server, SRF-DEPIR as follows. The algorithms $\overline{\mathsf{KeyGen}}^R, \overline{\mathsf{Prep}}^R, \overline{\mathsf{Query}}^R$ each simply emulate $\mathsf{KeyGen}, \mathsf{Prep}, \mathsf{Query}$ respectively. Whenever a query x is made to the crypto oracle, simply run the oracle-aided Turing machine $\mathcal{B}$ on x giving it oracle access to the secret random function R. When $\mathcal{B}^R(x)$ halts with a result y, reply to the query with y.

The stated efficiency follows by inspection. Correctness follows by the correctness of $(\mathsf{KeyGen}^{\mathcal{B}^R}, \mathsf{Prep}^{\mathcal{B}^R}, \mathsf{Query}^{\mathcal{B}^R})$ and since running the Turing machine $\mathcal{B}$ with the secret random function R as an oracle has the same distribution as the crypto oracle $\mathcal{B}^R$. Finally, for security, note that any adversary that breaks security of $(\overline{\mathsf{KeyGen}}^R, \overline{\mathsf{Prep}}^R, \overline{\mathsf{Query}}^R)$ is immediately also an adversary that breaks security of $(\mathsf{KeyGen}^{\mathcal{B}^R}, \mathsf{Prep}^{\mathcal{B}^R}, \mathsf{Query}^{\mathcal{B}^R})$ without even calling the crypto oracle. □

4.3 Discussion of Theorem 4.3

We note that a SK-DEPIR in the secret random function model implies a SK-DEPIR in the plain model that is secure assuming the existence of one-way functions. The client could simply sample a PRF key during KeyGen and store it as part of the secret key. Then throughout the scheme, any time one would query a value x to the secret random function, instead compute $\mathsf{PRF}(x)$. Security follows immediately from PRF security. This yields the following corollary.

Corollary 4.6. *Suppose SK-DEPIR exists (unconditionally) relative to a crypto oracle* $\mathcal{B}^R$*. Then there is a black-box construction of SK-DEPIR from one-way functions.*

The compilation preserves parameters in the same way as in Theorem 4.3.

On the other hand, a SK-DEPIR in the secret random function model also implies an unconditionally secure SK-DEPIR (without any oracles) if we allow the client to store an exponentially long secret key. Let L be the length of the longest query made to the secret random function oracle. Then during KeyGen the client simply samples a random function $R : \{0,1\}^{\leq L} \to \{0,1\}$ and stores it as part of the secret key. Whenever the original scheme would query the secret random function, simply look up the corresponding value from the key. This yields the following corollary.

Corollary 4.7. *Suppose SK-DEPIR exists (unconditionally) relative to a crypto oracle* $\mathcal{B}^R$ *with* δ*-security. Then there is a SK-DEPIR with exponentially long secret key that is* δ*-secure against computationally unbounded adversaries.*

The compiled scheme has the same round complexity, correctness and locality as in Theorem 4.3.

Finally, observe that we can apply Lemma 2.5 and combine with the results of Section 3 to show that any black-box constructions of SK-DEPIR from a broad class of cryptographic primitives yield all of the results of this section.

Corollary 4.8. *Let* $\mathcal{P}$ *be any primitive that exists relative to a crypto oracle. Suppose there is a black-box construction of SK-DEPIR from* $\mathcal{P}$*. Then the conclusions of Theorem 4.3 and Corollaries 4.6 and 4.7 hold.*

We note that while we use the phrase "black-box" in the formal sense (see Definition 2.3), the above result captures even some constructions of SK-DEPIR using what are often considered non-black-box techniques. For example, as we showed in Theorem 3.2, a combined "obfuscator/one-way function" primitive exists relative to a crypto oracle. Typically constructions that use obfuscation together with one-way functions are non-black-box as they depend on the code of the one-way function. However, those constructions are frequently black-box in the "combined primitive" of an obfuscator that can obfuscate programs that use the one-way function in a black-box way.

Remark 4.9 (On computational assumptions)**.** The statement of Theorem 4.3 only formally refers to compiling an SK-DEPIR that exists unconditionally relative to a crypto oracle (that is, it is secure against unbounded-time, poly-query adversaries, see Definition 2.2) into one in the secret random function model. However, our compiler is actually slightly more general in that it is agnostic to the existence of additional computational assumptions. That is, if there is some SK-DEPIR that can be constructed in the idealized world of a crypto oracle that is secure against PPT adversaries under some additional cryptographic assumption, then our compiler can remove the crypto oracle yielding a (plain-model) SK-DEPIR that is secure under the same cryptographic assumption (and perhaps also one way functions). For example, if one were to construct SK-DEPIR assuming the DDH assumption holds and using an idealized VBB obfuscator, then our compiler shows that one might as well have constructed SK-DEPIR only assuming DDH holds. In the language of [CFM21], our results show that, for any cryptographic primitive $\mathcal{P}$ that exists relative to a crypto oracle, $\mathcal{P}$ is *black-box useless* for constructing SK-DEPIR in the presence of any cryptographic assumption that implies one-way functions.

Remark 4.10 (On semi black-box constructions)**.** The definition we gave of a black-box construction in Definition 2.3 is often called fully black-box in the literature (see [RTV04]). It requires both that the implementation is black-box in the underlying primitive and that the reduction is black-box in the primitive and the adversary. A weaker notion that is also often considered is that of a semi black-box construction where the reduction is allowed to depend in a non-black-box way on the adversary. One key difference is that Lemma 2.5 does not naturally extend to the semi black-box setting. However, Reingold, Trevisan and Vadhan show that if $\mathcal{P}$ satisfies an additional property that it *allows embedding*, then they can prove version of Lemma 2.5 (see [RTV04, Definition 3.4 and Theorem 3.5]) for semi black-box constructions (and even the slightly weaker notion of $\forall\exists$-semi black-box constructions). As it turns out, this additional property is a mild constraint, and most cryptographic primitives, including obfuscation and all manners of advanced encryption, do allow embedding.

Let $\mathcal{P}$ be any cryptographic primitive that exists relative to a crypto oracle and allows embedding. Using the [RTV04] lemma, a semi black-box construction of SK-DEPIR from $\mathcal{P}$ would imply a SK-DEPIR that is secure against PPT adversaries exists relative to a crypto oracle. Then, applying our compiler, this in turn would imply that there is a semi black-box construction of SK-DEPIR from one-way functions.

Remark 4.11 (Efficiency)**.** Since our main focus is preserving locality in the compilers that we define, Theorem 4.3 doesn't explicitly state how our compiler impacts the run time of the various algorithms. However we note that our compiler has relatively little impact. In terms of offline running time, it does not affect the running time of Prep or the size of the preprocessed database. As for online time, Lemma 4.4 preserves the total running time of Π but moves all of the work

onto the client; and Lemma 4.5 only incurs a multiplicative overhead in $T_{\mathcal{B}}$, the running time of the Turing machine that implements the crypto oracle.

5 Ruling Out 2-Round Passive-Server SK-DEPIR

In Section 4 we showed that any SK-DEPIR that is built relative to even very strong idealized cryptography implies one in the much weaker secret random function model (in particular, it implies an SK-DEPIR from one-way functions, see Corollary 4.6). With this in mind, in this section, our goal is to fully rule out SRF-DEPIR. Unfortunately, we are only able to achieve our goal in a special case. Specifically, we rule out 2-round, passive-server SRF-DEPIR.

Theorem 5.1. *Let $\delta(\lambda)$ be a negligible function, $\varepsilon(\lambda) \leq 1/20$ and $\ell(\lambda, N) = o(N) + \mathrm{poly}(\lambda)$. There does not exist a 2-round, passive-server SRF-DEPIR with ε-correctness, δ-security (against unbounded adversaries), and locality ℓ.*

We prove the above theorem over the remainder of this section, and we provide a high level overview below. But first, observe that combining Theorem 5.1 with Theorem 4.3 and Lemma 2.5 yields the following.

Corollary 5.2. *A 2-round, passive-server SK-DEPIR with correctness, security and locality as in Theorem 5.1 does not exist relative to any crypto oracle (see Definition 2.2).*

Additionally, let $\mathcal{P}$ be any primitive that exists relative to a crypto oracle. There is no black-box construction (see Definition 2.3) of 2-round, passive-server SK-DEPIR from $\mathcal{P}$ with parameters as above.

Notation. Since we will only focus on the 2-round, passive-server setting throughout this section, we modify our DEPIR notation for the sake of convenience. In this setting, the interactive client query algorithm can be thought of as consisting of two separate (non-interactive) algorithms. We will write the algorithms of a DEPIR as $(\mathsf{KeyGen}, \mathsf{Prep}, \mathsf{Query}, \mathsf{Dec})$, where Query outputs a set of locations to be requested from the (passive) server and an intermediate state; and Dec decodes the database value from the server's response and the state from Query.

Proof Overview of Theorem 5.1. As we observed in Section 4.3, one can view a SRF-DEPIR as a SK-DEPIR that is secure against unbounded adversaries, but where the secret-key is exponentially large. In a prior work Canetti, Holmgren and Richelson (CHR) [CHR17] actually already rule out constructing SK-DEPIR that is secure against unbounded adversaries. Roughly, they show that any 2-round passive-server SK-DEPIR with sublinear locality can only be secure against (computationally unbounded) adversaries that only get to observe B-many queries where B is roughly the length of the secret key. This is not good enough for us, because the SRF corresponds to an exponentially large secret key.

To get around this, we notice that the only property that the CHR result really needs is for the key used during Query to have small mutual information with the preprocessed database. With that in mind, we define another variant of SK-DEPIR that we call "split-key" SK-DEPIR and show how to build it from SRF-DEPIR.

In the next section we define split-key SK-DEPIR formally. Later we prove Lemmas 5.5 and 5.6 which respectively show how to compile 2-round passive-server SRF-DEPIR into split-key SK-DEPIR, and adapt the CHR result to rule out split-key SK-DEPIR. Theorem 5.1 follows directly from combining the two lemmas.

Remark 5.3 (Ruling out semi black-box constructions)**.** Following the thread of Remark 4.10, we again remark that our results extend to semi black-box constructions. Let $\mathcal{P}$ be any cryptographic primitive that exists (unconditionally) relative to a crypto oracle $\mathcal{B}^R$ and allows embedding. Combining our compiler from Theorem 4.3 with the results of this section shows that there is an oracle, namely $\mathcal{B}^R$, relative to which $\mathcal{P}$ exists and is secure against all polynomial-query adversaries, but there is a (inefficient) polynomial-query attacker $\mathcal{A}^{\mathcal{B}^R}$ that breaks any SK-DEPIR.

However, following the framework of [RTV04], in order to rule out a semi black-box construction of SK-DEPIR from $\mathcal{P}$, we need an oracle relative to which $\mathcal{P}$ exists and there is a *efficient* attacker on any SK-DEPIR. To achieve this, simply add the inefficient attacker itself to the oracle, yielding $\mathcal{B}^R, \mathcal{A}^{\mathcal{B}^R}$. Clearly querying $\mathcal{A}$ gives an efficient attack on any SK-DEPIR, and $\mathcal{P}$ still exists relative to this new oracle because any polynomial-query adversary here could be implemented relative to just $\mathcal{B}^R$ by just running $\mathcal{A}$.

5.1 Definition of Split-key SK-DEPIR

The KeyGen algorithm of a split-key SK-DEPIR generates three keys $(\mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}})$, and each of Prep, Query, and Dec uses a different subset of keys. We will think of $\mathsf{sk}_{\mathrm{prp}}$ and $\mathsf{sk}_{\mathrm{rnd}}$ as each being (exponentially) long independent randomness, and $\mathsf{sk}_{\mathrm{qry}}$ as being a shorter (i.e. polynomial in length) random string that can be correlated with $\mathsf{sk}_{\mathrm{prp}}$ but does not depend on $\mathsf{sk}_{\mathrm{rnd}}$. One can think of split-key DEPIR as a weakening of SRF-DEPIR because $\mathsf{sk}_{\mathrm{prp}}$ and $\mathsf{sk}_{\mathrm{rnd}}$ can be thought of as both being derived from disjoint subdomains of a random function, and we restrict which subdomains are accessible in each part of the protocol.

The definition is similar to standard DEPIR (Definition 2.1), where we augment the syntax to handle the three keys.

Definition 5.4 (Split-key SK-DEPIR)**.** *A (2-round passive-server)* split-key SK-DEPIR *is a SK-DEPIR (Definition 2.1) with the following modified syntax.*

- $(\mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}) \leftarrow \mathsf{KeyGen}(1^\lambda, N)$*: Outputs the secret key in three parts* $\mathsf{sk}_{\mathrm{prp}}$*,* $\mathsf{sk}_{\mathrm{qry}}$*, and* $\mathsf{sk}_{\mathrm{rnd}}$*.*
- $\widetilde{\mathsf{DB}} \leftarrow \mathsf{Prep}(\mathsf{sk}_{\mathrm{prp}}, \mathsf{DB})$*: it gets only* $\mathsf{sk}_{\mathrm{prp}}$ *and a database* DB*, and it outputs the preprocessed database* $\widetilde{\mathsf{DB}}$*.*
- $(I, \mathsf{st}) \leftarrow \mathsf{Query}(\mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i)$*: it gets keys* $(\mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}})$*, index* i *outputs a set* I *and state* st*.*
- $b \leftarrow \mathsf{Dec}(\mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, \mathsf{st}, i, a)$*: it gets all of the keys* $(\mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}})$*, the state* st *from* Query*, an index* i *and the server's response* a*; and it outputs a bit* b*.*

The algorithms satisfy the following properties.

ε-Correctness and δ-Security. *These are the same as in Definition 2.1, making the mechanical modifications to reflect the modified syntax. Also, for security, we consider computationally unbounded adversaries that query the challenger a polynomial number of times (strictly speaking, δ is a function of both the security parameter and the number of queries).*

Non-correlation. *Over the random coins of* KeyGen, $\mathsf{sk}_{\mathrm{rnd}}$ *is independent from* $(\mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{qry}})$*.*

Efficiency. *The keys* $\mathsf{sk}_{\mathrm{prp}}$ *and* $\mathsf{sk}_{\mathrm{rnd}}$ *may be exponentially large, but we require* $|\mathsf{sk}_{\mathrm{qry}}|$ *to be bounded by some* $\mathrm{poly}(\lambda, N)$*. We model all the algorithms as having random access to their inputs, and we require all the algorithms to run in time* $\mathrm{poly}(\lambda, N)$*, except for* KeyGen *(which may sample exponentially long keys). We define locality as in Definition 2.1.*

5.2 Split the Key in SRF-DEPIR

Given any 2-round passive-server SK-DEPIR in the secret random function model, we show how to transform it into a split-key scheme. The transformation extracts a short query key $\mathsf{sk}_{\mathrm{qry}}$ from the secret random function, so that the Query algorithm can be performed correctly using $\mathsf{sk}_{\mathrm{qry}}$ instead of the random function R. Our observation is that the computation of Query is *independent of* DB so we can *emulate* Query *during the key generation* and extract a small but heavy subset in R by concentration. Then we can include the heavy set as (part of) the short $\mathsf{sk}_{\mathrm{qry}}$ enabling queries.

Transformation to split-key DEPIR. Let $(\mathsf{KeyGen}^R_{\mathsf{srf}}, \mathsf{Prep}^R_{\mathsf{srf}}, \mathsf{Query}^R_{\mathsf{srf}}, \mathsf{Dec}^R_{\mathsf{srf}})$ be a 2-round passive-server SRF-DEPIR. Let $m := m(N)$ be a number of repetitions that will be determined later, and let $\tilde{N} := \tilde{N}(N)$ be the length of the preprocessed database specified by $\mathsf{Prep}_{\mathsf{srf}}$.

$\mathsf{KeyGen}(1^\lambda, N)$**:** Sample the three keys as follows.

1. Sample a random function R and use it to sample $k_{\mathsf{srf}} \leftarrow \mathsf{KeyGen}^R_{\mathsf{srf}}(1^\lambda, N)$.
2. Initialize an empty set Q. For each $i \in [\![N]\!]$, repeat the following m times:
   - Emulate $\mathsf{Query}^R_{\mathsf{srf}}(k_{\mathsf{srf}}, i)$ using the sampled function R (but use a fresh internal random tape each time). Whenever $\mathsf{Query}_{\mathsf{srf}}$ calls $R(z)$ for some z, add z to Q.
3. Let $R|_Q$ denote the restriction of R to Q (i.e., $R|_Q := \{(z, R(z)) : z \in Q\}$).
4. Sample an independent random function R', and sample a one-time pad $K \leftarrow \{0,1\}^{\tilde{N}}$ independently and uniformly at random.
5. Set $\mathsf{sk}_{\mathrm{prp}} = (R, k_{\mathsf{srf}}, K)$, $\mathsf{sk}_{\mathrm{qry}} = (R|_Q, k_{\mathsf{srf}})$ and $\mathsf{sk}_{\mathrm{rnd}} = R'$ and output $(\mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}})$.

$\mathsf{Prep}(\mathsf{sk}_{\mathrm{prp}}, \mathsf{DB})$**:** Parse $\mathsf{sk}_{\mathrm{prp}} = (R, k_{\mathsf{srf}}, K)$, then compute and output $\widetilde{\mathsf{DB}} := \mathsf{Prep}^R_{\mathsf{srf}}(k_{\mathsf{srf}}, \mathsf{DB}) \oplus K$.

$\mathsf{Query}(\mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i)$**:** Parse $\mathsf{sk}_{\mathrm{qry}} = (R|_Q, k_{\mathsf{srf}})$ and $\mathsf{sk}_{\mathrm{rnd}} = R'$.

1. Let $\tilde{R}$ denote the function
$$\tilde{R}(z) := \begin{cases} R_Q(z) & \text{if } z \in Q \\ R'(z) & \text{otherwise.} \end{cases}$$
2. Use $\tilde{R}$ to run $(I, \mathsf{st}_{\mathsf{srf}}) \leftarrow \mathsf{Query}^{\tilde{R}}_{\mathsf{srf}}(k_{\mathsf{srf}}, i)$, and let $\tilde{Q}$ be the set of all of the queries $\mathsf{Query}_{\mathsf{srf}}$ makes to $\tilde{R}$.
3. Output the set I along with state $\mathsf{st} := (\tilde{Q}, \mathsf{st}_{\mathsf{srf}})$.

$\mathsf{Dec}(\mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, \mathsf{st}, i, a)$**:** Parse $\mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}$, and st as defined above.

1. Let R^* denote the function

$$R^*(z) := \begin{cases} R'(z) & \text{if } z \in \tilde{Q} \setminus Q \\ R(z) & \text{otherwise.} \end{cases}$$

2. Run and output $b \leftarrow \mathsf{Dec}_{\mathsf{srf}}^{R^*}(k_{\mathsf{srf}}, \mathsf{st}_{\mathsf{srf}}, i, a \oplus K[I])$ using R^*, where $K[I]$ denotes the sequence $(K[j] : j \in I)$ (recall that the server's response is $a = \widetilde{\mathsf{DB}}[I]$).

Lemma 5.5. *Let $p(\lambda, N)$ be a polynomial. Suppose $(\mathsf{KeyGen}_{\mathsf{srf}}, \mathsf{Prep}_{\mathsf{srf}}, \mathsf{Query}_{\mathsf{srf}}, \mathsf{Dec}_{\mathsf{srf}})$ is a 2-round passive-server SRF-DEPIR with δ-security, locality ℓ and ε-correctness. Then there exists a polynomial $m(\lambda, N)$ (depending on p) such the above construction with m repetitions yields a (2-round passive-server) split-key SK-DEPIR with δ-security, locality ℓ, and $(\varepsilon + 1/p)$-correctness.*

Proof. The syntax and locality of the scheme can be verified mechanically. Next we argue security. Let $\mathsf{DB} \in \{0,1\}^N$, let $B = \mathrm{poly}(\lambda)$, let $(j_1, j_2, \ldots, j_B)$ be an arbitrary sequence of locations in $[\![\widetilde{N}]\!]$, and let $(i_0, i_1) \in [\![\widetilde{N}]\!]^2$ be the challenge. For $b \in \{0,1\}$, let $\mathbf{H}_b$ denote the distribution

$$\mathbf{H}_b := \left(\widetilde{\mathsf{DB}}, I_b, J_1, J_2, \ldots, J_B\right)$$

over the probability space defined by the random coins of KeyGen and independent executions of $(I_b, \mathsf{st}_b) \leftarrow \mathsf{Query}(\mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i_b)$ and $(J_k, \mathsf{st}'_k) \leftarrow \mathsf{Query}(\mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, j_k)$ for all $k \in [B]$. It suffices to show that $\mathbf{H}_0$ and $\mathbf{H}_1$ are statistically δ-close (i.e. $\mathbf{H}_0 \approx_\delta \mathbf{H}_1$).

Since $\widetilde{\mathsf{DB}}$ is masked by a one-time pad, the joint distribution is identical to a hybrid distribution that samples the preprocessed database uniformly at random,

$$\mathbf{H}_b \equiv (U, I_b, J_1, J_2, \ldots, J_B)\,,$$

where U is a uniform and independent random bit string of length $\tilde{N}$. Also by construction the queries output by Query are the same as those output by $\mathsf{Query}_{\mathsf{srf}}^{\tilde{R}}$ and $\tilde{R}$ is uniformly random and independent of U. Thus by δ-security of the SRF-DEPIR we have

$$(U, I_0, J_1, J_2, \ldots, J_B) \approx_\delta (U, I_1, J_1, J_2, \ldots, J_B)\,,$$

which completes the proof of security.

It remains to show ε-correctness. Let $\lambda \in \mathbb{N}$, $\mathsf{DB} \in \{0,1\}^N$, $i \in [\![N]\!]$ and consider the experiment

$$\begin{aligned} (\mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}) &\leftarrow \mathsf{KeyGen}(1^\lambda, N) \\ \widetilde{\mathsf{DB}} &\leftarrow \mathsf{Prep}(\mathsf{sk}_{\mathrm{prp}}, \mathsf{DB}) \\ (I, \mathsf{st}) &\leftarrow \mathsf{Query}(\mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i). \end{aligned}$$

In the above experiment, let Q be the set of calls to R made by $\mathsf{Query}_{\mathsf{srf}}^{R}$ observed by KeyGen, let P be the set of calls to R made by $\mathsf{Prep}_{\mathsf{srf}}^{R}$ observed by Prep, and let $\tilde{Q}$ be the set of calls to $\tilde{R}$ made by $\mathsf{Query}_{\mathsf{srf}}^{\tilde{R}}$ observed by Query.

We define $\mathsf{CONSISTENT}$ to denote the event that in the execution of Query, $\tilde{R}$ is never called on an input z that was called during Prep, but is not in Q. That is, this is the event that $(\tilde{Q} \setminus Q)$ is disjoint from P.

Conditioned on CONSISTENT occuring, the entire correctness experiment can be viewed as always using R^* that is identical to R, which perfectly emulates the SRF-DEPIR and thus is correct with probability ε. Now it suffices to show that $\Pr[\mathsf{CONSISTENT}] \geq 1 - 1/p(\lambda, N)$.

Let $\ell_{\mathsf{p}} := \ell_{\mathsf{p}}(\lambda, N)$ be the number of calls made to R during $\mathsf{Prep}^R_{\mathsf{srf}}$, and let $\ell_{\mathsf{q}} := \ell_{\mathsf{q}}(\lambda, N)$ be the number of calls made to R during $\mathsf{Query}^R_{\mathsf{srf}}$. Let random variables $X_1, X_2, \ldots, X_{\ell_{\mathsf{q}}}$ be the sequence of calls to $\tilde{R}$ for the given i (i.e., the jth call to $\tilde{R}$ is $\tilde{R}(X_j)$). We will show that for each j, $X_j \notin P \vee X_j \in Q$ with high probability. Since $X_j \in \tilde{Q}$ by definition, we will obtain the desired claim. The detailed calculation is given below.

We begin by working under the simplifying assumption that $R' = R$, hence all algorithms use the same oracle R, but we will remove this assumption later. Let $\alpha > 0$ be a threshold probability to be determined later. If it holds that $\Pr[X_j \in P] < \alpha$, we already finished a case. Otherwise, we have $\beta := \Pr[X_j \in P] \geq \alpha$. Consider the identical but independently sampled random variable Z_j that is the jth call to R during an execution of $\mathsf{Query}^R_{\mathsf{srf}}(\mathsf{sk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i)$ with fresh randomness. Then, we have

$$\Pr[X_j = Z_j] \geq \Pr[X_j = Z_j \mid E] \cdot \Pr[E] \geq \frac{1}{\ell_{\mathsf{p}}} \cdot \beta^2 \geq \frac{\alpha^2}{\ell_{\mathsf{p}}}$$

where E denotes the event $X_j \in P \wedge Z_j \in P$, which happens with probability β^2, and the second inequality follows by the collision probability in P and the fact that $|P| \leq \ell_{\mathsf{p}}$.

Now we set $m := N \cdot \ell_{\mathsf{p}}/\alpha^2$ and consider m independent and identical samples $Z_j^{(k)}$ for $k \in [m]$. The probability that X_j is in Q is at least the probability that X_j is equal to one of the $Z_j^{(k)}$'s. Thus we have

$$\Pr[X_j \in Q] \geq 1 - \left(1 - \frac{\alpha^2}{\ell_p}\right)^{N \cdot \ell_{\mathsf{p}}/\alpha^2} \geq 1 - e^{-N} \geq 1 - \alpha$$

for any $\alpha = 1/\mathrm{poly}(N)$ and all sufficiently large N. It holds that $\Pr[X_j \notin Q \wedge X_j \in P] < \alpha$ in both cases, and by a union bound,

$$\Pr[\exists j, X_j \notin Q \wedge X_j \in P] < \alpha \cdot \ell_{\mathsf{q}}.$$

The above event is exactly the negation of $(\tilde{Q} \setminus Q) \cap P = \emptyset$. Now we dispense with our simplifying assumption, because in the real experiment the oracle $\tilde{R}$ is defined using $R|_Q$ and an independent R' instead of the same R as in KeyGen and Prep.

We eliminate the assumption by a hybrid argument. Let $\tilde{R}_j$ be the following hybrid function,

$$\tilde{R}_j(z) := \begin{cases} R_Q(z) & \text{if } z \in Q \\ R(z) & \text{else if } z \text{ is the } j'\text{th call to } \tilde{R}_j \text{ for } j' \leq j \\ R'(z) & \text{otherwise.} \end{cases}$$

We have $\tilde{R}_0 = \tilde{R}$ and $\tilde{R}_{\ell_{\mathsf{q}}} = R$. Consider the hybrid experiment that uses $\tilde{R}_j$ during the execution of Query and let A_j be the event $\tilde{Q} \setminus Q \cap P = \emptyset$ in this hybrid. We have $\Pr[A_{\ell_{\mathsf{q}}}] \geq 1 - \alpha \cdot \ell_{\mathsf{q}}$ but want to bound $\Pr[A_0]$. Observe that A_j and A_{j-1} differ only in the "bad event" $X_j \notin Q \wedge X_j \in P$; otherwise, the distributions $R(X_j)$ and $R'(X_j)$ are identical and so are the distributions A_j and A_{j-1}. Since the experiment have not yet called R' before X_j, we have X_j depends only on R, and by the same analysis, the bad event happens with probability at most α. It follows that A_j and A_{j-1} are α-close statistically, and thus $\Pr[A_0] \geq 1 - 2\alpha \cdot \ell_{\mathsf{q}}$ by hybrid lemma.

Given that $p, \ell_{\mathsf{p}}, \ell_{\mathsf{q}}$ are polynomials in N, it suffices to choose $\alpha := 1/\mathrm{poly}(N)$ for a sufficiently large $\mathrm{poly}(N) \geq 2p(N) \cdot \ell_{\mathsf{q}}(N)$ and thus sufficiently large $m = N \cdot \ell_{\mathsf{p}}/\alpha^2$.

This completes the proof of Lemma 5.5. □

5.3 Ruling Out Split-key SK-DEPIR

In this section we give the second step of the proof of Theorem 5.1, ruling out the existence of (2-round passive-server) split-key SK-DEPIR. More formally, we prove the following.

Lemma 5.6. *Let $\delta(\lambda)$ be a negligible function, $\varepsilon(\lambda) \leq 1/10$, and $\ell(\lambda, N) = o(N) + \mathrm{poly}(\lambda)$. There does not exist a 2-round passive-server split-key SK-DEPIR with ε-correctness, δ-security (against unbounded adversaries), and locality ℓ.*

To prove the above lemma, we adapt the proof of Canetti, Holmgren, and Richelson (CHR) [CHR17, Theorem 3] to handle the setting of split-key SK-DEPIR. The adapted result gives a lower bound on the size of $\mathsf{sk}_{\mathrm{qry}}$ for any scheme that would be secure against an a priori fixed bounded number of queries.

Definition 5.7 (Bounded-query Security)**.** *A split-key SK-DEPIR is B-*bounded-query δ-secure *if there exists an a priori bound $B = \mathrm{poly}(\lambda)$ such that the scheme is secure against all adversaries that only get to observe B executions of the query protocol in the security game.*

We contrast the above definition with that which we have considered throughout this paper that states only that for every adversary there must exist some arbitrarily large polynomial bound on the number of executions the adversary observes. We are now ready to state the more technical lemma that is adapted directly from [CHR17].

Lemma 5.8. *In any split-key SK-DEPIR, that is ε-correct and B-bounded-query δ_{sk}-secure, the key size $|\mathsf{sk}_{\mathrm{qry}}|$ satisfies:*

$$|\mathsf{sk}_{\mathrm{qry}}| \geq \left(1 - 2B \cdot \delta_{sk} - H\left(\frac{1-\delta_{pk}}{2}\right)\right) \cdot B - O(\log B),$$

where H is the binary entropy function, δ_{pk} is the smallest δ such that $H(\varepsilon + \delta) \geq 1 - \frac{\ell + O(\log N)}{N}$, where ℓ is the locality of the scheme, and N is the database size. In particular, if $\ell = o(N), \varepsilon \leq 1/10$, and $\delta_{sk} \leq 1/(20 \cdot B)$, then $|\mathsf{sk}_{\mathrm{qry}}| = \Omega(B)$.

Lemma 5.8 immediately implies Lemma 5.6 because the fixed size of $\mathsf{sk}_{\mathrm{qry}}$ sampled by KeyGen immediately puts a polynomial bound on the number of queries B the adversary can see, ruling out unbounded-query security for $\delta = \delta_{\mathsf{sk}} = \mathrm{negl}(\lambda)$. Combining Lemma 5.6 with Lemma 5.5 setting $p(\lambda, N) \geq 20$ immediately yields Theorem 5.1.

The proof of Lemma 5.8 is obtained by augmenting the proof of Canetti, Holmgren, and Richelson (CHR) [CHR17, Theorem 3]. The main difference is that in a split-key scheme, the correctness requires all three keys whereas the public-key security reveals only one key; nonetheless, our proof relies on a $\mathsf{Reconstruct}$ algorithm that takes two keys, and the analysis relies on both correctness and security. The main difference between our result here and [CHR17, Theorem 3] are colored in blue.

In the proof, $\Delta(x_1, x_2)$ denotes the Hamming distance between strings (x_1, x_2), $H(\alpha)$ denotes the binary entropy of probability α, and for random variables X, Y, $H(X)$ denotes the entropy of X, $I(X;Y)$ denotes the mutual information, and $\mathsf{SD}(X;Y)$ denotes the statistical difference. We henceforth borrow the proof of CHR and lightly modify it when necessary.

5.3.1 Locality in Split-key PK-DEPIR

First, we define a "public-key" variant of split-key SK-DEPIR as follows.

Definition 5.9. *A* split-key public-key DEPIR (PK-DEPIR) *is a split-key SK-DEPIR where security holds against adversaries that are given* $\mathsf{sk}_{\mathrm{qry}}$ *as input. Since* $\mathsf{sk}_{\mathrm{qry}}$ *is the only piece of information that is necessary to perform queries, in the security game, there is no need to provide the sequence of learning queries (Step 2 of Definition 2.1).*

We show that a split-key PK-DEPIR cannot achieve both information theoretic security and non-trivial succinctness. The intuition is that (by statistical security) a single query is indistinguishable from a query to any other index, so the server's answer contains almost all the information about the database. Let ℓ denote the locality of the scheme.

Lemma 5.10. *In any statistically* δ*-secure split-key PK-DEPIR with correctness error* ε*, it holds that*

$$\ell \geq (1 - H(\varepsilon + \delta))N - O(\log N).$$

Proof. Suppose the split-key PK-DEPIR is δ-secure. Then, we claim the following.

Claim 5.10.1. *There is an unbounded algorithm* Reconstruct *such that for any database* DB*,*

$$\mathbb{E}[\Delta(\mathsf{DB}', \mathsf{DB})] \geq (1 - \varepsilon - \delta)N$$

in the probability space defined by sampling

$$\begin{aligned}
(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{rnd}}) &\leftarrow \mathsf{KeyGen}(1^\lambda, N) \\
\widetilde{\mathsf{DB}} &\leftarrow \mathsf{Prep}(\mathsf{sk}_{\mathrm{prp}}, \mathsf{DB}) \\
q &\leftarrow \mathsf{Query}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, 0) \\
a &:= \widetilde{\mathsf{DB}}[q] \\
\mathsf{DB}' &\leftarrow \mathsf{Reconstruct}(\mathsf{sk}_{\mathrm{prp}}, \mathsf{pk}_{\mathrm{qry}}, q, a).
\end{aligned}$$

Proof. Reconstruct is an algorithm which does the following:

1. Given $(\mathsf{sk}_{\mathrm{prp}}, \mathsf{pk}_{\mathrm{qry}}, q, a)$, it samples $(\mathsf{sk}_{\mathrm{rnd}1}, r_1), \ldots, (\mathsf{sk}_{\mathrm{rnd}N}, r_N)$, where each $(\mathsf{sk}_{\mathrm{rnd}i}, r_i)$ is sampled independently from the distribution of random strings that satisfy the condition $q = q' \leftarrow \mathsf{Query}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}i}, i; r_i)$.

   Henceforth, we denote $\hat{r}_i := (\mathsf{sk}_{\mathrm{rnd}i}, r_i)$ for short.

2. Finally, it outputs DB' such that for each i, $\mathsf{DB}'_i = \mathsf{Dec}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{prp}}, \hat{r}_i, i, a)$.

By linearity of expectation, it suffices to show that for a uniformly random i,

$$\Pr[\mathsf{DB}_i = \mathsf{Dec}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{prp}}, \hat{r}_i, i, a)] \geq 1 - \varepsilon - \delta.$$

Recall that the δ-security of split-key requires that

$$\left(\mathsf{pk}_{\mathrm{qry}}, \mathsf{DB}, \widetilde{\mathsf{DB}}, q_0 := \mathsf{Query}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, 0)\right) \approx_\delta \left(\mathsf{pk}_{\mathrm{qry}}, \mathsf{DB}, \widetilde{\mathsf{DB}}, q_i := \mathsf{Query}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i)\right).$$

Because q is a function of $\mathsf{pk}_{\mathrm{qry}}$ and independent variables, the distribution $(\mathsf{pk}_{\mathrm{qry}}, \mathsf{DB}, \widetilde{\mathsf{DB}}, q, \mathsf{sk}_{\mathrm{prp}})$ is identical to $(\mathsf{pk}_{\mathrm{qry}}, \mathsf{DB}, \widetilde{\mathsf{DB}}, q, \mathsf{sk}_{\mathrm{prp}}{}')$, where $\mathsf{sk}_{\mathrm{prp}}{}'$ is independently sampled from $(\mathsf{sk}_{\mathrm{prp}}{}', \mathsf{pk}_{\mathrm{qry}}{}', \mathsf{sk}_{\mathrm{rnd}}{}') \leftarrow \mathsf{KeyGen}$ conditioned on $\widetilde{\mathsf{DB}} = \mathsf{Prep}(\mathsf{sk}_{\mathrm{prp}}{}', \mathsf{DB})$ and $\mathsf{pk}_{\mathrm{qry}} = \mathsf{pk}_{\mathrm{qry}}{}'$. Hence, by δ-security, it holds that

$$\left(\mathsf{pk}_{\mathrm{qry}}, \mathsf{DB}, \widetilde{\mathsf{DB}}, q_0, \mathsf{sk}_{\mathrm{prp}}\right) \approx_\delta \left(\mathsf{pk}_{\mathrm{qry}}, \mathsf{DB}, \widetilde{\mathsf{DB}}, q_i, \mathsf{sk}_{\mathrm{prp}}\right). \tag{3}$$

Suppose we instead sample $q \leftarrow \mathsf{Query}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i)$ in the claim. This change modifies the distribution of $(\mathsf{pk}_{\mathrm{qry}}, \mathsf{DB}, \widetilde{\mathsf{DB}}, q, \mathsf{sk}_{\mathrm{prp}})$ by at most δ by Equation (3). Because a is a function of $(\widetilde{\mathsf{DB}}, q)$ and $\hat{r}_i$ is computed from $(\mathsf{pk}_{\mathrm{qry}}, q)$ (as in Reconstruct), the distribution of $(\mathsf{DB}, \mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{prp}}, \hat{r}_i, i, a)$ and therefore the probability that $\mathsf{DB}[i] = \mathsf{Dec}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{prp}}, \hat{r}_i, i, a)$ also changes by at most δ. However, the modified experiment induces a distribution of $(\mathsf{DB}, \mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, q, r)$ which is exactly as if one had sampled $q \leftarrow \mathsf{Query}(\mathsf{pk}_{\mathrm{qry}}, \hat{r}_i, i)$. Thus in the modified experiment,

$$\Pr[\mathsf{DB}[i] = \mathsf{Dec}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{prp}}, \hat{r}_i, i, a)] \geq 1 - \varepsilon,$$

which means that in the original experiment

$$\Pr[\mathsf{DB}[i] = \mathsf{Dec}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{prp}}, \hat{r}_i, i, a)] \geq 1 - \varepsilon - \delta.$$

□

Lemma 5.10 will also use the following.

Claim 5.10.2 (Claim 4 of CHR [CHR17]). *A random variable $X \in \{0,1\}^N$ with expected Hamming weight at most εN (with $\varepsilon \leq 1/2$) has entropy at most $H(\varepsilon)N + O(\log N)$.*

To prove the lemma, we use the following basic fact. If X and Y are random variables, then

$$0 \leq H(X) - H(X|Y) \leq H(Y).$$

We apply this where $X = \mathsf{DB}$ is uniformly random (i.e. has entropy N), and $Y = a$. The fact that Reconstruct produces DB' which agrees in expectation with DB on at least $(1 - \varepsilon - \delta)N$ locations implies that $H(\mathsf{DB}|a) \leq H(\varepsilon + \delta)N + O(\log N)$. Thus

$$\ell \geq H(a) \geq N - (H(\varepsilon + \delta)N + O(\log N)) = (1 - H(\varepsilon + \delta))N - O(\log N),$$

which concludes the proof of Lemma 5.10. □

5.3.2 Key Size for Split-key SK-DEPIR

The idea behind our lower bound on key size for a secret-key scheme is that releasing the key turns the scheme into a public-key scheme, which by Lemma 5.10 must be insecure. Since the key is a piece of information which causes a dramatic change in the entropy of a random variable, it must be long.

Proof of Lemma 5.8. Given any split-key SK-DEPIR scheme with correctness error ε, locality ℓ, and B-query δ_{sk}-security, we can turn it into a public-key scheme with the same correctness probability and locality ℓ by just releasing the secret key. By Lemma 5.10, this scheme can only be δ-secure if

$$H(\varepsilon + \delta) \geq 1 - \frac{\ell + O(\log N)}{N}.$$

Let δ_{pk} denote the minimum such δ. Note that if $\varepsilon = o(1)$ and $\ell = o(N)$, then $\delta_{pk} = \frac{1}{2} - o(1)$. By definition of δ_{pk}, there exists a database DB and indices i_0 and i_1 such that the distributions $(\mathsf{pk}_{\mathrm{qry}}, \widetilde{\mathsf{DB}}, q_0)$ and $(\mathsf{pk}_{\mathrm{qry}}, \widetilde{\mathsf{DB}}, q_1)$ are δ_{pk}-far (i.e. distinguishable with probability $\frac{1+\delta_{pk}}{2}$) in the probability space defined by sampling

$$\begin{aligned}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{prp}}, \mathsf{sk}_{\mathrm{rnd}}) &\leftarrow \mathsf{KeyGen}(1^\lambda, N)\\ \widetilde{\mathsf{DB}} &\leftarrow \mathsf{Prep}(\mathsf{sk}_{\mathrm{rnd}}, \mathsf{DB})\\ q_0 &\leftarrow \mathsf{Query}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i_0)\\ q_1 &\leftarrow \mathsf{Query}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i_1).\end{aligned}$$

In particular, suppose a uniformly random string $M = (m_1, \ldots, m_B) \in \{0,1\}^B$ is chosen, and an adversary $\mathcal{A}$ is given $(\mathsf{pk}_{\mathrm{qry}}, \widetilde{\mathsf{DB}}, q_1, \ldots, q_B)$, where $q_j \leftarrow \mathsf{Query}(\mathsf{pk}_{\mathrm{qry}}, \mathsf{sk}_{\mathrm{rnd}}, i_{m_j})$. Then it is possible for $\mathcal{A}$ to output an M' such that in expectation, $\Delta(M, M') \leq \frac{1-\delta_{pk}}{2} \cdot B$. In other words, by Claim 5.10.2,

$$H\left(M \mid (\mathsf{pk}_{\mathrm{qry}}, \widetilde{\mathsf{DB}}, q_1, \ldots, q_B)\right) \leq H\left(\frac{1-\delta_{pk}}{2}\right) B + O(\log B). \tag{4}$$

On the other hand, we can use the following lemma, proved implicitly by Bellare, Tessaro, and Vardy [BTV12] and explicitly stated by Dodis [Dod12] to lower bound the entropy of M given only $\mathsf{DB}, q_1, \ldots, q_B$.

Claim 5.10.3 (Bellare, Tessaro, and Vardy)**.** *For any (possibly correlated) distributions M, C over some spaces $\mathcal{M}$ and $\mathcal{C}$, let*

$$\varepsilon = \mathsf{SD}((M, C); M \times C)$$

where $M \times C$ is the product distribution of the independent marginal distributions M and C. Then,

$$2\varepsilon^2 \leq I(M; C) \leq 2\varepsilon \cdot \log(|\mathcal{M}|/\varepsilon).$$

Let $C := (\widetilde{\mathsf{DB}}, q_1, ..., q_B)$. The B-bounded-query δ_{sk}-security of our split-key SK-DEPIR implies that $\mathsf{SD}((M, C); M \times C) \leq \min(1, B \cdot \delta_{sk})$ (by a standard hybrid argument over B queries). Let $\delta'_{\mathsf{sk}} = \min(1, B \cdot \delta_{sk})$. By Claim 5.10.3, we have

$$\begin{aligned}H(M|\widetilde{\mathsf{DB}}, q_1, ..., q_B) &= H(M|C)\\ &= H(M) - I(M; C)\\ &\geq B - 2\delta'_{sk}(B - \log \delta'_{sk})\\ &= (1 - 2\delta'_{sk})B + 2\delta'_{sk} \log \delta'_{sk}\\ &> (1 - 2\delta'_{sk})B - 2.\end{aligned} \tag{5}$$

$|\mathsf{pk}_{\mathrm{qry}}| = |\mathsf{sk}_{\mathrm{qry}}|$ must be large as the difference between these two entropies (Equations (4) and (5)), which proves Lemma 5.8. That is

$$\begin{aligned}|\mathsf{sk}_{\mathrm{qry}}| &\geq (1-2\delta'_{sk})B - 2 - H\left(\frac{1-\delta_{pk}}{2}\right)B - O(\log B)\\ &= \left(1 - 2\delta'_{sk} - H\left(\frac{1-\delta_{pk}}{2}\right)\right)B - O(\log B).\end{aligned}$$

As a special case, if $\ell = o(N)$, $\varepsilon \leq 1/10$, and $\delta'_{sk} \leq 1/20$, then $\delta_{pk} \in [0.4, 0.5]$ and $H((1-\delta_{pk})/2) \in [0.81, 0.89]$ and thus $|\mathsf{sk}_{\mathrm{qry}}| = \Omega(B)$.

This completes the proof of Lemma 5.8.

□

References

[BHMW21] Elette Boyle, Justin Holmgren, Fermi Ma, and Mor Weiss. On the security of doubly efficient PIR. Cryptology ePrint Archive, Report 2021/1113, 2021. 1

[BHW19] Elette Boyle, Justin Holmgren, and Mor Weiss. Permuted puzzles and cryptographic hardness. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 465–493. Springer, Cham, December 2019. 1

[BIM00] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 55–73. Springer, Berlin, Heidelberg, August 2000. 1, 5

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 662–693. Springer, Cham, November 2017. 1, 2, 5

[BTV12] Mihir Bellare, Stefano Tessaro, and Alexander Vardy. Semantic security for the wiretap channel. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 294–311. Springer, Berlin, Heidelberg, August 2012. 24

[CFM21] Geoffroy Couteau, Pooya Farshim, and Mohammad Mahmoody. Black-box uselessness: Composing separations in cryptography. In James R. Lee, editor, *ITCS 2021: 12th Innovations in Theoretical Computer Science Conference*, volume 185, pages 47:1–47:20. LIPIcs, January 2021. 15

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50. IEEE Computer Society Press, October 1995. 1

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 694–726. Springer, Cham, November 2017. 1, 5, 16, 21, 23

[DH24] Jesko Dujmovic and Mohammad Hajiabadi. Lower-bounds on public-key operations in PIR. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 65–87. Springer, 2024. 3

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 122–138. Springer, Berlin, Heidelberg, May 2000. 2

[DMZ23] Xiuquan Ding, Giulio Malavolta, and Tianwei Zhang. Doubly efficient batched private information retrieval. Cryptology ePrint Archive, Report 2023/1552, 2023. 13

[Dod12] Yevgeniy Dodis. Shannon impossibility, revisited. In Adam Smith, editor, *ICITS 12: 6th International Conference on Information Theoretic Security*, volume 7412 of *Lecture Notes in Computer Science*, pages 100–110. Springer, Berlin, Heidelberg, August 2012. 24

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 445–456. Springer, Berlin, Heidelberg, February 2005. 2

[KO00] Eyal Kushilevitz and Rafail Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 104–121. Springer, Berlin, Heidelberg, May 2000. 1

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Barna Saha and Rocco A. Servedio, editors, *55th Annual ACM Symposium on Theory of Computing*, pages 595–608. ACM Press, June 2023. 1, 5, 6

[LR86] Michael Luby and Charles Rackoff. How to construct pseudo-random permutations from pseudo-random functions (abstract). In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, page 447. Springer, Berlin, Heidelberg, August 1986. 11

[MMN+16] Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and abhi shelat. Lower bounds on assumptions behind indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 49–66. Springer, Berlin, Heidelberg, January 2016. 7

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Berlin, Heidelberg, February 2004. 7, 15, 17