Lower Bounds for PIR with Preprocessing from Blackbox Cryptography

Alexander Hoover [*] Giuseppe Persiano [†] Kevin Yeo [‡]

Abstract

Single-server private information retrieval (PIR) schemes are known to require linear query time. Recent works circumvent these classical lower bounds by leveraging preprocessing to answer queries in sublinear time. We prove computation lower bounds for PIR with preprocessing schemes making blackbox usage of any cryptography (such as random oracles or virtual blackbox obfuscation). If the client stores s bits about an n-bit database, then answering $k = \Omega(s)$ queries requires either $\Omega(n/s)$ amortized online communication or $\Omega(n/s)$ amortized server cryptographic operations. This is tight, as known constructions match either bound while outperforming the other. Our lower bound is unconditional and allows arbitrary query protocols, weakened privacy, and server-side database encodings (including doubly efficient PIR) whenever the encoding is independent of the blackbox cryptography. Previous bounds were only known conditionally and for restricted classes of preprocessing, e.g., under non-encoding assumptions. Our framework also yields $\Omega(n/s)$ communication lower bounds for schemes with $o(n/s)$ server cryptographic operations, communication-determined server cryptography, or perfect privacy in the idealized model. Finally, we prove lower bounds for symmetric PIR with client preprocessing in the random oracle model and give a matching construction using only one-way functions in the online phase.

[*]Stevens Institute of Technology, Email: ahoover@stevens.edu
[†]Università di Salerno, Email: giuper@gmail.com
[‡]Google, Email: kwlyeo@google.com

Contents

1 Introduction **1**
- 1.1 Our Contributions . . . 2
- 1.2 Our Techniques . . . 5
  - 1.2.1 Computation Lower Bounds . . . 6
  - 1.2.2 Communication Lower Bounds . . . 11
  - 1.2.3 Symmetric PIR without Online Public-Key Cryptography . . . 12

2 Related works **12**

3 Preliminaries **14**
- 3.1 Crypto Oracles . . . 16
- 3.2 PIR with Preprocessing . . . 18
- 3.3 Subkey prediction . . . 20
- 3.4 Blackbox constructions . . . 21

4 Computation Lower Bounds **22**
- 4.1 Dual Private Information Retrieval . . . 23
- 4.2 Constructing a Dual PIR . . . 24
- 4.3 Correctness Analysis of Dual PIR Construction . . . 28
- 4.4 Impossibility of Dual PIR . . . 37
- 4.5 Impossibility of (Public-Key) Doubly Efficient PIR . . . 40
- 4.6 Impossibility of Weak PIR (with Preprocessing) . . . 42

5 Communication Lower Bounds **43**
- 5.1 PIR with Sublinear Server Cryptographic Operations . . . 43
- 5.2 PIR with Communication-Determined Server Cryptographic Operations . . . 44
- 5.3 PIR with Perfect Privacy . . . 47

6 Symmetric PIR without Online Public-Key Cryptography **49**
- 6.1 Formal model . . . 49
- 6.2 Limitations of SPIR without Public-Key Cryptography . . . 51
- 6.3 Toy Constructions . . . 54
- 6.4 SPIR with Sublinear Online Computation . . . 55
- 6.5 Detailed Construction . . . 57

A Communication Costs for Few Queries **69**

B Generalization to Resampleable Oracles **69**

1 Introduction

Private Information Retrieval (PIR) is a very powerful cryptographic primitive that enables a client to retrieve the i-th entry from a n-bit database $D \in \{0,1\}^n$ held by a server with the privacy guarantee that the server cannot learn the index i queried by the client. Leveraging its strong privacy guarantees, PIR is a core building block for a wide range of privacy enhancing technologies including advertising [GLM16], blocklist checks [KC21], contact discovery [DRRT18, KRS+19], distributed file systems [MVM24], leaked password check [TPY+19, LPA+19, ALP+21], private communication [ACLS18] and web search [HDCGZ23]. In recent years, PIR has also seen real-world deployments in industry including caller ID [App24], device enrollment [YP21] and password leak detection [LKLM21]. Given the importance of PIR, there has been substantial work in studying its complexity and efficiency.

PIR has been studied in two main settings: the multiple-server setting where the database is held by two or more servers that are non-colluding, introduced by Chor, Kushilevitz, Goldreich and Sudan [CGKS95], and the single-server setting put forth by Kushilevitz and Ostrovsky [KO97]. It is known that any single-server PIR with sublinear communication implies oblivious transfer [DMO00] that cannot be built from blackbox OWFs [IR89] and the server must perform a linear number of public-key cryptography [DH24]. In contrast, there exists two-server PIR with $\mathrm{polylog}(\lambda, n)$ communication using only one-way functions (OWFs) [CG97, BGI16] as well as $n^{o(1)}$ communication without any cryptographic assumptions at all [DG15]. Even though multi-server PIR constructions are known to be more efficient, their privacy guarantees are more fragile as they critically depend on stronger non-collusion assumptions between different parties that are challenging to obtain in practice. For this reason, the study of single-server PIR is important and it will be the focus of our work.

PIR with Client Preprocessing. As a way to improve the efficiency of PIR, there has been a recent line of work that studies PIR with client preprocessing where the database is preprocessed by the client in an offline phase and the information gathered is used to improve the efficiency of online queries (see [PPY18, CK20, KC21, SACM21, CHK22, MZRA22, Yeo23b, LP23b, Yeo23a, LP23a, GZS24, LP24] and references therein). These constructions obtained sublinear query time circumventing known linear query time lower bounds for PIR without preprocessing [BIM00, DH24]. Surprisingly, it turns out the preprocessing model also enables building single-server PIR from weaker assumptions. Recent works, starting from Piano [ZPZS24] and follow-ups [RMS24, HPPY25, WR25], have shown it is possible to build single-server PIR with sublinear query communication from blackbox OWFs in the client preprocessing model. In more detail, these works show that, if a client stores s bits about an n-bit database, then PIR schemes require only $\tilde{O}(n/s)$ (amortized) communication and computation per query. In a standard parameter setting with client storage $s = O(\sqrt{n})$, the query computation and communication becomes $\tilde{O}(\sqrt{n})$. Furthermore, there exist schemes [ZPZS24, RMS24, HPPY25, WR25] where the server performs no cryptography during queries (contrasting the linear public key operations lower bound in [DH24]).

In terms of lower bounds for PIR with client preprocessing model, the known limitations for constructions apply only to special cases. The results in [CK20, CHK22, Yeo23b] prove computation lower bounds, but require very strong non-encoding assumptions for the server that must store the n-bit database without any modification for responding to queries. Recent work [ISW24] showed barriers for communication-efficient schemes restricted to database-oblivious and single-roundtrip query constructions, where all client requests may depend only on the queried index

and the randomness (and not on the database). In particular, such a construction would imply a separation between the complexity classes SZK and BPP, which remains an open complexity theory problem for many natural settings including the random oracle model. Therefore, it remains open to unconditionally resolve the best possible efficiency achievable by a single-server PIR with client preprocessing.

PIR with Server Preprocessing. We note that there also exists a line of work studying PIR with server preprocessing where all the additional information is stored publicly by the server [BIM00, BIPW17, CHR17, HOWW19, PY22] that is also known as doubly efficient PIR (DEPIR). A recent breakthrough work presented a single-server DEPIR using $\tilde{O}(1)$ query communication and time assuming the hardness of Ring LWE [LMW23]. In this model, it is already known that public-key cryptography is necessary to obtain sublinear query communication (as observed in [HPPY25]). Furthermore, it was proven that restricted classes of DEPIR with single-roundtrip queries and passive servers acting as memory (without any computation) are impossible from blackbox cryptography [LMW25]. However, it remains unknown whether this impossibility applies to all possible DEPIR schemes from blackbox cryptography without any other restrictions.

Symmetric PIR with Client Preprocessing. We also extend the problem to try and understand what additional properties could be achieved by single-server PIR with preprocessing when depending only on one-way functions. For example, we could consider the stronger notion of symmetric PIR [GIKM98] where we also wish to guarantee that the client may retrieve at most one database entry per query from the server. By definition, it is known that symmetric PIR (without preprocessing) implies oblivious transfer even if the symmetric PIR is inefficient and uses linear (or even more) communication [DMO00]. One can wonder whether it is possible to circumvent this blackbox impossibility in the client preprocessing model. Given the above, it is clear that it is insufficient to depend only on one-way functions in the entire construction. However, it remains plausible to build a SPIR with client preprocessing that relies only upon one-way functions during query time.

1.1 Our Contributions

As our main result, we prove lower bounds for single-server PIR with preprocessing making blackbox usage of cryptography. Throughout our work, we will use PIR with preprocessing to encompass both client preprocessing and server preprocessing (unless otherwise specified). To model blackbox cryptography usage, we suppose schemes have access to a crypto oracle [DH24, LMW25] that may be used to implement a wide-range of ideal powerful cryptographic primitives, including random oracles, generic multilinear groups and virtual blackbox obfuscation (see Section 3.1 for more details). All our lower bounds will focus on the efficiency during online query time and are agnostic to the cost of the offline phase (used to compute the client storage and/or server encodings). In particular, our results apply to all constructions whose offline phase proceeds in two steps. First, the server is able to perform a one-time encoding of the database. Afterwards, the client and server perform joint computation to enable the client to obtain some private state. We note this encompasses a wide range of PIR with preprocessing models including PIR with client preprocessing and public-key DEPIR. Our lower bounds apply to all constructions where the first server database encoding step is performed without any oracle queries (or no database encoding is performed at all). To our knowledge, this aligns with all blackbox constructions known to date (as well as state-of-the-art non-blackbox constructions too). All other steps of

the PIR protocol are able to have unrestricted oracle access. See Definition 3.8 for more details on oracle-free database encodings.

Computation Lower Bounds. We present computation lower bounds showing that any single-server PIR with client preprocessing that uses any blackbox cryptography (in the crypto oracle model) cannot be more efficient than currently known constructions.

Theorem 1.1 (Informal)**.** *There does not exist a single-server PIR with client preprocessing scheme making blackbox usage of cryptography for an n-bit database with s-bit client storage supporting a single batch query of $k = \Theta(s)$ indices with amortized online query computation of $t = o(n/s)$. In particular, we show each such single-server PIR scheme must satisfy at least one of the following:*

- *The amortized online query communication must be $\Omega(n/s)$ bits.*
- *The amortized server cryptographic bit operations must be $\Omega(n/s)$.*

Our result is tight in multiple ways. First, there exist schemes that achieve exactly $t = O(n/s)$ online query time [WR25] for an unbounded number of queries using OWF as a blackbox. Secondly, there exist schemes that bypass exactly one of the above requirements on the batch size, computation, and communication. For small batch sizes, there exist single-server PIR schemes with $o(n/s)$ online communication for a bounded number of less than s queries (such as [ZLTS23, LP23a, GZS24]). To obtain small communication, these schemes may be combined with blackbox fully homomorphic encryption (FHE) techniques to support unbounded queries by privately refreshing the s-bit client storage using $o(n)$ communication [FLLP24]. However, all these FHE approaches require amortized $\Omega(n/s)$ FHE operations to refresh the s-bit client storage (even when ignoring the communication from the original offline phase). For server computation, there exists schemes [ZPZS24, HPPY25] supporting unbounded queries with $\tilde{O}(n/s)$ amortized online communication where the server performs no cryptographic operations. In other words, our lower bound identifies two fundamental bottlenecks such that any scheme will inevitably suffer from at least one of them.

Furthermore, our results are more widely applicable than prior lower bounds. In particular, our lower bounds hold for any single-server PIR with client preprocessing scheme that makes blackbox usage of cryptography, requiring only that the server's database encoding is computed without crypto oracle queries (the client's private state may still be computed using the crypto oracle). In contrast, the server time lower bounds in [CK20, CHK22, Yeo23b] only apply to schemes with server non-encoding assumptions such that the server must use the database without modification for queries. Note that our restriction is considerably weaker than non-encoding: the server may still encode the database arbitrarily, so long as the encoding is computed without querying the crypto oracle. Additionally, the communication (and, thus, computation) lower bounds in [ISW24] only apply to database-oblivious, single-roundtrip query constructions where all client requests depend only on the queried index and internal randomness (along with the lower bound requiring additional complexity assumptions). Finally, we note that the lower bounds in [DH24] may be interpreted into the preprocessing model showing that the total time across the offline and online phases must be linear. In contrast, our lower bounds show that the amortized online query time must be $\Omega(n/s)$ regardless of the cost of the offline phase (including even if the client's offline phase performs a linear number of public-key operations).

Our techniques also give lower bounds for doubly efficient PIR (DEPIR) with server preprocessing. In particular, we rule out the existence of DEPIR built from blackbox cryptography.

Theorem 1.2 (Informal)**.** *There does not exist a single-server doubly efficient PIR making blackbox usage of cryptography for an n-bit database with query computation of $t = o(n)$.*

As a caveat, our lower bound only applies to public-key DEPIR whereas the prior impossibility [LMW25] also applies to secret-key DEPIR where the server's encoding may depend on a client private state as studied in [BIPW17, CHR17, CIMR25]. Our lower bound places no restriction on the query protocol, which may consist of arbitrarily many rounds with an actively computing server. In contrast, the prior impossibility in [LMW25] only applied to DEPIR with single-roundtrip query algorithms with a passive server restricted to acting only as memory and performing no computation, while allowing the preprocessing to use the oracle (unlike our result). Therefore, the two results are incomparable. Nevertheless, our oracle-free database encoding restriction aligns with the current state-of-the-art public-key DEPIR [LMW23], even though it is a non-blackbox construction. See Section 4.5 for further discussion.

Finally, we note that our lower bounds also apply to PIR schemes that provide privacy guarantees against weaker adversaries that are restricted to a fixed polynomial number of crypto oracle queries (similar to weak key exchange from Merkle puzzles).

Theorem 1.3 (Informal)**.** *There does not exist a weak single-server PIR with preprocessing scheme for an n-bit database with s-bit client storage supporting a single batch query of $k = \Theta(s)$ indices with amortized online query computation of $t = o(n/s)$ that makes q crypto oracle queries and provides privacy against adversaries that make at most $O(q)$ crypto oracle queries.*

The above lower bound may also be applied to single-server PIR (without preprocessing) implying there does not exist any such weak schemes that obtain sublinear online query time against weak adversaries that make the same number of crypto oracle queries as an honest execution of the PIR protocol. To our knowledge, this was not previously known. The prior blackbox impossibility shows sublinear communication PIR implies oblivious transfer [DMO00, DH24], but there are known constructions of weak oblivious transfer that are secure against such restricted adversaries [BGI08].

Communication Lower Bounds. We also prove communication lower bounds for certain classes of single-server PIR with preprocessing schemes that match current constructions.

Theorem 1.4 (Informal)**.** *There does not exist a single-server PIR with preprocessing scheme making blackbox usage of cryptography for an n-bit database with s-bit client storage supporting a batch query of $k = \Theta(s)$ indices with amortized online communication of $o(n/s)$ if the construction satisfies at least one of the following requirements:*

- *The amortized server cryptographic bit operations is $o(n/s)$.*
- *All server cryptographic operations depend only on the communication transcript.*
- *Perfect privacy (in an idealized model).*

For the first two settings, we require the server's database encoding to be computed without crypto oracle queries. Our perfect privacy result, however, requires no such server's database encoding restriction.

The first setting is a direct corollary of our computation lower bound, which is tight as there exist schemes with $\tilde{O}(n/s)$ communication where the server performs no cryptography [ZPZS24, HPPY25]. For the second setting, we consider PIR schemes where the server's cryptographic operations depend only on the communication transcript. To our knowledge, all OWF-based

single-server PIR schemes satisfy this criterion and, furthermore, there exist OWF-based schemes with $O(n/s)$ communication exactly matching our lower bound [WR25]. Additionally, there do exist schemes with cryptographic operations that depend on the database (not appearing in the communication) that achieve $o(n/s)$ communication using FHE [FLLP24]. Finally, we prove an $\Omega(n/s)$ communication lower bound for perfect privacy matched by known constructions [RMS24, HPPY25] when adapted to the random oracle model. This lower bound applies to schemes that derive their client query randomness from the random oracle and key material in their state, as done by the adapted constructions of [RMS24, HPPY25]. Notably, this is our one lower bound that requires no restriction on the server's database encoding.

Our lower bounds substantially improve over the only existing communication lower bound in this model [ISW24] that is only established under certain complexity-theoretic assumptions, and therefore yields only a barrier for blackbox usage of specific crypto oracles. Moreover, this bound only applies to certain restricted classes of PIR by making non-encoding and round assumptions. While our first two communication bounds assume an oracle-free database encoding, they permit arbitrary (oracle-free) encodings otherwise, as well as multi-round and database-dependent queries. See Section 2 for a detailed comparison.

Symmetric PIR with Client Preprocessing. We show that it is possible to obtain even stronger properties while still only relying upon OWFs during online queries. In particular, we present a construction of symmetric single-server PIR with client preprocessing where it is guaranteed that the client only obtains the queried entry and no other information about the database.

Theorem 1.5 (Informal)**.** *There exists a symmetric single-server PIR with client preprocessing where the query algorithm only relies upon OWFs using s-bit client storage with query communication $\tilde{O}(n/s)$ and query time $\tilde{O}(n/s)$ supporting $\tilde{O}(s)$ queries per offline phase.*

We note that the query communication costs of our construction match many prior works that only depend on OWFs during queries [ZPZS24, RMS24, HPPY25, WR25] without the additional symmetric privacy guarantees and similarly require no cryptographic server operations. Furthermore, the computational times match known lower bounds in the preprocessing model [CK20, CHK22, Yeo23b]. One may notice that our construction is weaker than previous schemes in that it only supports $\tilde{O}(s)$ queries per offline phase whereas prior non-symmetric PIR schemes can handle an unbounded polynomial number of queries. This turns out to be a limitation of symmetric PIR with preprocessing as we show:

Theorem 1.6 (Informal)**.** *In the random oracle model, there does not exist a symmetric single-server PIR with client preprocessing where the query algorithm only relies upon OWFs using s-bit client storage that supports $k > s$ queries per offline phase.*

Our symmetric PIR construction is tight in the number of supported queries up to $\tilde{O}(1)$ factors. Note, the offline phase of our construction relies upon public-key cryptography. However, this is necessary since symmetric PIR implies oblivious transfer that cannot be built only from blackbox OWFs [IR89]. If we insist on the online queries to be efficient and avoid public-key cryptography, then the offline phase must rely upon public-key cryptography (as done in our construction).

1.2 Our Techniques

We first quickly overview three types of lower bound techniques used in prior work to highlight where our techniques and results differ. The line of lower bounds in [CK20, CHK22, PY22, Yeo23b]

prove time lower bounds via compression arguments where a random database may be compressed using only a subset of database entries that are accessed by the server during online queries. However, this approach inherently requires a strong server non-encoding assumption for the compression method to identify the exact database entries accessed during the query algorithm.

The communication lower bound in [ISW24] takes a different approach by constructing PIR privacy adversaries assuming oracle access to a SZK-complete problem. However, this inherently means the lower bound is conditional on an additional complexity assumption along with imposing additional restrictions such as database-obliviousness and single-round queries like prior works [LV16].

Finally, recent works [DH24, LMW25] consider schemes using crypto oracles encapsulating blackbox cryptography usage. These lower bounds show that one can compile out all server crypto oracle usages in single-server PIR schemes to obtain a new construction that effectively only uses a random oracle. Afterwards, they rely on the fact that such a random oracle construction is impossible. While our lower bound techniques will borrow the crypto oracle modeling, the overall framework is not compatible with our goals as there do exist OWF-based single-server PIR with preprocessing schemes. Instead, we will develop new techniques to directly prove efficiency lower bounds.

1.2.1 Computation Lower Bounds

We start by presenting our computation lower bounds. We will consider single-server PIR with preprocessing schemes with access to a crypto oracle O: $\Pi^{\mathsf{O}} = (\mathsf{Init}, \mathsf{Setup}^{\mathsf{O}}, \mathsf{Query}^{\mathsf{O}})$. The Init algorithm enables the server to preprocess the n-bit database in polynomial time to produce an encoded database E for the server. In our model, Init is a plain randomized algorithm that makes no queries to the crypto oracle O (an *oracle-free encoding*), whereas Setup and Query have full oracle access; we explain below why our proof requires the encoding to be independent of O. The Setup algorithm receives the encoded database E and outputs a s-bit storage st for the client. As a note, this could be a two-party protocol between the client and server, but we assume the server observes nothing from this offline phase (as we are proving a lower bound, this only strengthens the result). The Query algorithm is an interactive two-party protocol between the client and the server. The client receives the $k = \Theta(s)$ indices of queries Q along with the s-bit client storage st while the server receives the encoded database E. We denote the execution of the query algorithm as $(\mathsf{ans}; \perp \mid \tau) \leftarrow \mathsf{Query}^{\mathsf{O}}(\mathsf{st}, Q; E)$ where the client receives the PIR's output as ans, the server receives no output and both parties observe the communication transcript τ.

Our goal is to prove a lower bound of $t = \Omega(n/s)$ amortized online query computation across $k = \Theta(s)$ queries. Equivalently, we want to show that the total computation during the execution of the query algorithm is $\Omega(n)$. In fact, we will prove that either the total communication is $\Omega(n)$ bits or the server queries the crypto oracle over $\Omega(n)$ bits during the execution of the query algorithm. For the sake of contradiction, we will suppose the existence of a scheme Π^{O} that is more efficient with both $o(n)$ bit communication and the server queries the crypto oracles on $o(n)$ bits during the query algorithm. We note that we make no assumptions about the efficiency of the setup or server initialization algorithms. Thus, our results will lower bound the online computational cost. The one structural assumption we make is that Init performs no crypto oracle queries. We note our results extend to encodings that query O on a sublinear total number of bits (by recording Init's queries in the dual PIR's advice, as in our sublinear-server-cryptography

setting) or whose oracle queries are determined by the communication transcript, though with additional caveats; see Remark 4.10.

Dual PIR. A central step in our proof is a reduction from PIR with preprocessing to a new notion called *dual PIR* that essentially inverts the order of the preprocessing and efficiency requirements of PIR. A dual PIR consists of three algorithms:

- $A \leftarrow \mathsf{Advice}(D)$: before any query indices are known, the client computes a t-bit *advice* A about the n-bit database D. We require the advice to be sublinear, $t = |A| = o(n)$, so that it cannot encode the entire database.
- $H \leftarrow \mathsf{Hint}(Q, D, A)$: given a set of k query indices Q, the database D, and the advice A, the client computes a s-bit *hint* H.
- $R \leftarrow \mathsf{Recon}(Q, A, H)$: using only the query set Q, the advice A, and the hint H — but, critically, *without* the database D — the client reconstructs the relevant entries $D[Q]$.

Reconstruction would be trivial if the hint H had room to write down $D[Q]$, i.e., when $s \geq k$. We are therefore interested in the limits of when a dual PIR can be correct once the hint is smaller than the number of queries, $s < k$.

The above is suggestively called dual PIR, because it intuitively inverts the efficiency requirements of standard PIR with preprocessing. The preprocessing step of dual PIR (without the query) uses $t = o(n)$ bits similar to the Query algorithm. The online portion of dual PIR with the query produces a s-bit encoding similar to the Setup algorithm. Furthermore, dual PIR has no privacy guarantees and is not even a two-party protocol. Instead, one can view it as a compression problem.

Using dual PIR, our proof will proceed in two steps. First, we show that when $t = o(n)$ and $s < k$, there cannot exist a dual PIR construction that is correct even with probability $2^{s-\Omega(k)}$ using a connection with leakage-resilient cryptography. Afterwards, we show that our too-efficient-to-be-true PIR with preprocessing Π^{O} may be used to construct such an impossible dual PIR.

Connection with Leakage-Resilient Cryptography. To prove that there cannot exist dual PIR schemes that are correct (even with exponentially small probability), we make a connection with the well-studied area of leakage-resilient cryptography [AARR03, SKI03, DP08, ADW09, ADW10, ADN+10]. At a high level, this model considers the setting where an adversary successfully compromises a remote system, but it is infeasible for the adversary to download all the data in the system. To build cryptography resilient to leakage in this attack, one can consider a very large private key (say, 1 TB) of n bits where it is assumed the adversary obtains at most $t < n$ bits. For efficiency purposes, any cryptographic primitive built in this system will only use a subset of k-bits of the large n-bit key. Therefore, a core problem in leakage-resilient cryptography is *subkey prediction* that upper bounds the probability that a computationally unbounded adversary may correctly guess k random locations of a n-bit random key when the adversary may compute any t-bit leakage about the key [BKR16].

Interestingly, we show that a very good dual PIR (in terms of efficiency and correctness) implies a successful adversary for the subkey prediction problem in leakage-resilient cryptography. We will view the n-bit database in dual PIR as the n-bit private key in subkey prediction. Then, we will view the t-bit advice output of dual PIR as the t-bit leakage chosen by the subkey prediction adversary. Finally, when the subkey prediction adversary is given the k random locations Q of the n-bit key to guess, we will execute the final Recon step of dual PIR where we simply choose a s-bit

random string to replace the output of the dual PIR's second Hint step. If the underlying dual PIR was correct with probability ε, then the subkey prediction adversary is correct with probability at least $\varepsilon \cdot 2^{-s}$ corresponding to when the correct s-bit encoding output of Hint is guessed correctly.

We can now combine our subkey prediction adversary built from dual PIR with prior bounds proven on the best possible success probability for any subkey prediction adversary in [BKR16]. In particular, it was shown that for $t \leq n/2$ leakage and $k = \Omega(\log n)$ locations of interest, the adversary can predict the k bits with probability at most $2^{-\Omega(k)}$. This immediately implies that there does not exist any dual PIR that is correct with probability larger than $\varepsilon > 2^{s-\Omega(k)}$ unless the output of the Advice step in the dual PIR is $t \geq n/2$ bits. We note that the subkey prediction bound in [BKR16] is an information-theoretic result that holds for any computationally unbounded adversary with arbitrary shared randomness across the adversary's algorithms. Thus, the impossibility holds for dual PIR algorithms that are computationally inefficient with access to oracles.

Dual PIR Construction Overview. Next, we will construct a dual PIR using a too-efficient-to-be-true PIR with preprocessing scheme $\Pi^{\mathsf{O}} = (\mathsf{Init}, \mathsf{Setup}^{\mathsf{O}}, \mathsf{Query}^{\mathsf{O}})$ with an oracle-free database encoding. Our dual PIR will have access both to a crypto oracle O and a random oracle RO. Furthermore, our dual PIR will not necessarily be computationally efficient, but the above impossibility on the subkey prediction problem will still apply since it is an information-theoretic result. However, we will ensure our dual PIR makes at most a polynomial number of queries to the crypto oracle O in total as we will use our dual PIR as a tool to build a privacy adversary for the underlying PIR scheme (critical to proving correctness of our dual PIR). In contrast, our dual PIR may make exponentially many queries to the random oracle RO (for example, to sample the exponentially many candidate crypto oracles $\mathsf{O}_1, \ldots, \mathsf{O}_z$); this is harmless, since the dual PIR need not be efficient and only its polynomially many crypto oracle queries are inherited by the privacy adversary.

At a high level, we wish to use the strong privacy properties of the underlying PIR Π^{O} to build the dual PIR. In the first step, we can have $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}(D)$ execute Π^{O} properly for an arbitrary query Q' to obtain a communication transcript τ, which will be the t-bit advice $A = \tau$. Now, the second Hint algorithm receives the real queries Q. By privacy, then it should be that this communication transcript $A = \tau$ must also be consistent with the given query Q even if it is different than the original query $Q' \neq Q$ used in Advice. Then, Hint can iterate through all possible s-bit client storage that may be output by $\mathsf{st} \leftarrow \mathsf{Setup}^{\mathsf{O}}(E)$ of the underlying PIR scheme that would produce the communication transcript τ when executing $\mathsf{Query}^{\mathsf{O}}(\mathsf{st}, Q; E)$ where $E \leftarrow \mathsf{Init}(D)$ is the server's encoding of the database in the offline phase (recomputed by Hint using encoding randomness derived from the shared random oracle, so that all dual PIR algorithms agree on the same E). The output of the hint algorithm will be this s-bit client storage $H = \mathsf{st}$. Effectively, the second hint algorithm is re-randomizing the client of the underlying PIR while keeping the communication transcript τ the same. Finally, $\mathsf{Recon}^{\mathsf{O},\mathsf{RO}}(Q, A, H)$ will execute the client-side portion of the underlying PIR's query algorithm with the s-bit client storage discovered by the hint algorithm $H = \mathsf{st}$ using the communication transcript from advice $A = \tau$ where the server is simulated using the communication transcript.

While the above is the general framework behind our dual PIR construction, we note that there are many obstacles to concretely instantiate these ideas into a dual PIR scheme that is both provably correct and efficient in terms of crypto oracle queries (to apply PIR privacy later).

Sampling Client States. We start with the core challenge of our dual PIR scheme that is to sample

a s-bit client storage in the Hint algorithm that is consistent with the communication transcript τ output by Advice. A trivial approach is to try all the possible internal random coin tosses of the underlying PIR's Setup algorithm to obtain a candidate s-bit client storage and execute the underlying PIR's Query algorithm to check if the resulting communication transcript matches τ Note, this is possible since Hint receives all the necessary information including the query Q and database D. However, this would require an exponential number of queries to the crypto oracle O.

Instead, our Hint algorithm will sample new crypto oracles O' with the goal of finding a crypto oracle and s-bit client storage pair $(\mathsf{O}', \mathsf{st}')$ such that the execution of the underlying PIR's Query algorithm results in the desired communication transcript τ. Since the crypto oracle O' is locally sampled, we no longer care about the number of queries performed. Thus, we can now use the above approach of trying all internal random coin tosses to the PIR's Setup algorithm and check if the resulting s-bit client storage used in the execution of the PIR's Query algorithm results in a communication transcript τ. If no such s-bit client storage is found, we can continue to sample new crypto oracles O' until finding a successful tuple of s-bit client storage and crypto oracle O'.

While the above approach avoids too many queries to the crypto oracle O, it creates several new problems. First, Hint must also now encode the newly sampled crypto oracle O' to be used by Recon, which we will discuss in more detail later. An even more subtle (but critical) issue is that Recon can no longer discern whether a crypto oracle O' is compatible with the database D and communication transcript τ. In particular, Recon cannot determine whether the server's communication can match the transcript τ for the given crypto oracle O' since Recon does not receive the database D as input. As an example, the underlying PIR scheme could include a $\sqrt{n}$ bits consisting of XOR of random bits from the crypto oracle and the first $\sqrt{n}$ bits of the input database D. Now, a randomly sampled crypto oracle O' is very unlikely to be compatible with the input database D and communication transcript τ (with probability $2^{-\sqrt{n}}$). However, Recon has no way to figure this out since it does not have the database D.

Instead, we will modify our dual PIR construction such that the Hint algorithm only randomly samples crypto oracles O' such that the server's communication will always be consistent with communication transcript for the input database D. To do this, the Advice algorithm's t-bit output A will consist of both the communication transcript τ and of all the queries to the crypto oracle O performed by the server during the execution of the Query algorithm that we denote by the set S. As a reminder, we assumed towards a contradiction that the server of the underlying PIR queries the crypto oracle on $o(n)$ bits, meaning that $|S| = o(n)$ bits. As all algorithms of the dual PIR have access to the crypto oracle O, they can all compute the crypto oracle responses: $\mathsf{O}(S) = \{(q, \mathsf{O}(q)) \mid q \in S\}$. Now, the Hint algorithm will only randomly sample crypto oracles O' conditioned on $\mathsf{O}'(S) = \mathsf{O}(S)$. For any such sampled crypto oracle O', we note that the server's execution is now completely deterministic since all its crypto oracle responses are fixed (the server's other input, the encoded database E, is unchanged as well since Init makes no crypto oracle queries). Thus, the server's resulting communication using O' with input database D will always match the transcript τ generated using the original crypto oracle O. With this modification, our result will now lower bound the total size of the online communication and the total number of server-side crypto oracle queries (i.e., blackbox cryptographic operations).

Efficiently Encoding Crypto Oracles. It still remains to show that the Hint algorithm can efficiently encode the newly sampled crypto oracle using $O(s)$ bits as required by the dual PIR. To do this, we will use the random oracle RO that is shared across all the dual PIR algorithms. Using

RO, Hint will sample crypto oracles $\mathsf{O}_1, \ldots, \mathsf{O}_z$ such that $\mathsf{O}_i(S) = \mathsf{O}(S)$ for all $i \in [z]$. Furthermore, Recon can also re-sample the exact same crypto oracles using RO. A first attempt might be for Hint to find some index $i \in [z]$ that enables finding a s-bit client storage st that successfully matches the communication transcript τ and encode both st and the index $i \in [z]$ using $\log_2(z)$ bits. However, one may need to try as many as $z = 2^{\Omega(|\tau|)}$ different sampled crypto oracles before finding one that successfully matches the communication transcript τ such as if the communication transcript distribution is uniformly random. Note, this is too large as encoding the index requires $\log_2(z) = \Omega(|\tau|)$ bits, and $|\tau|$ may be much larger than s, so it will not fit in the s-bit hint.

Instead, we design our Hint algorithm to take a different approach. For convenience, we suppose each sampled crypto oracle O_i is also associated with a uniformly random s-bit string st_i and client randomness $\omega_{\mathsf{Client},i}$ generated using the random oracle RO so that it is shared across both Hint and Recon. For each $i \in [z]$ in increasing order, Hint will do two things. First, it checks whether st_i is a feasible output the underlying PIR's Setup algorithm for some internal random coin tosses. Secondly, Hint checks whether the execution of the query algorithm $\mathsf{Query}^{\mathsf{O}_i}(Q, \mathsf{st}_i; E)$ (using $\omega_{\mathsf{Client},i}$ for client randomness) matches the communication transcript τ, where $E \leftarrow \mathsf{Init}(D)$ is the recomputed encoded database. The Hint algorithm will maintain a counter c tracking the number of times that a pair $(\mathsf{O}_i, \mathsf{st}_i)$ ends up matching the communication transcript τ (that is, every time the second check succeeds). For the first time when both checks performed by the Hint algorithm succeeds, the output of the Hint algorithm will be the current counter value of the counter, $H = c$. In other words, the output of Hint is the index c such that the c-th pair $(\mathsf{O}_i, \mathsf{st}_i)$ that reproduces the communication transcript τ under the underlying PIR's Query algorithm is the first one whose st_i is also a valid s-bit client storage output of the underlying PIR's Setup algorithm. This counter $H = c$ will be passed to Recon that does not receive the database D. It turns out that Recon can check whether $(\mathsf{O}_i, \mathsf{st}_i)$ generates transcript τ (even without D) as we will show later. However, Recon cannot check if st_i is output of Setup that requires D, but will use this counter instead. Finally, we also show that this counter may always be encoded using at most $3s$ bits except with 2^{-s} probability.

Recon will iterate through all z sampled crypto oracle and client storage pairs $(\mathsf{O}_i, \mathsf{st}_i)$ and perform the second check as done in Hint. We note that Recon does not have the database D as input, so it cannot execute the full Query algorithm as done in Hint. Instead, Recon will only execute the client-side of the Query algorithm using the server's communication in the transcript τ to simulate the server's algorithm that requires the database D. We critically leverage the fact that each crypto oracle O_i is sampled to match $\mathsf{O}(S)$ here. Due to this restriction, we know that the server's responses will always match the transcript τ. As a result, the client-side simulation of the server during Recon is performed correctly and will match the execution performed during Hint. To produce the final answer, Recon finds the c-th instance of the communication transcripts matching and returns the underlying PIR's output when executing the client-side of the Query algorithm.

Proving Correctness of Our Dual PIR. Our correctness proof for the dual PIR proceeds in two steps. First, we show that the dual PIR is correct when the arbitrary query Q' used in Advice is also given as input to the other steps of the dual PIR, Hint and Recon. To prove this is correct, we show that the output distribution of our dual PIR is identical to the output distribution of a proper execution of the underlying PIR using Q' (except when the Hint algorithm fails). Our proof shows that the dual PIR samples from the same distribution as the underlying PIR in an inverted order: sampling communication transcripts first and client storage conditioned on the transcript.

As the encoded database E is then independent of O, a crypto oracle resampled conditioned only on agreeing with O on the server's query set S is distributed exactly as the real oracle conditioned on everything the dual PIR has fixed (namely, E and $\mathsf{O}(S)$).

In the second step, we prove correctness of our dual PIR even when the Hint and Recon algorithms receive any query $Q \neq Q'$ different from Q' used in Advice. If our dual PIR is incorrect for some $Q \neq Q'$, we build a privacy adversary that distinguishes between transcripts generated from Q and those from Q'. At a high level, the adversary executes the last two steps of the dual PIR with the challenge transcript. As the adversary chooses the database D, the adversary may check whether the dual PIR's output is correct. The adversary successfully distinguishes transcripts from the two query sequences by observing the difference in correctness between Q and Q'. Note that our PIR adversary makes a polynomial number of crypto oracle queries since its main cost is executing the underlying dual PIR. In fact, our PIR adversary makes the same number of crypto oracle queries as an honest server makes during the Query protocol of the underlying PIR. As a note, our adversary may lazily sample a local random oracle that may be used the dual PIR (as it is not used by the underlying PIR). Thus, our lower bound even applies to weaker PIR schemes provide privacy only against adversaries with a fixed polynomial number of queries.

1.2.2 Communication Lower Bounds

Small Server Cryptographic Operations. We note this setting is a direct corollary of our above computational lower bound. As a reminder, the t-bit advice in the dual PIR consists of both the communication transcript and all server crypto oracle queries. If the server makes crypto oracle queries on at most $o(n/s)$ amortized bits (or $o(n)$ total bits), this immediately implies that the communication must be $\Omega(n)$ bits in total or $\Omega(n/s)$ amortized per retrieval.

Communication-Dependent Server Cryptographic Operations. We consider the setting where all the server crypto oracle queries depend only on the communication transcript. To our knowledge, this matches all OWF-based PIR schemes (in some cases trivially, when the server performs no cryptographic operations). We modify our dual PIR construction to omit all the server crypto oracle queries S in the t-bit output of Advice. Instead, the latter two steps of our dual PIR, Hint and Recon, may recompute the server crypto oracle queries S using the communication transcript and the crypto oracle O. So we obtain a $\Omega(n/s)$ amortized communication lower bound.

PIR with Perfect Privacy. Next, we consider schemes with perfect privacy that derive their client query randomness from the random oracle and a seed in their state (making the online query algorithm deterministic for a fixed state and oracle). At first glance, this category of PIR may seem overly restrictive; however, our bound applies to any such perfectly private PIR relative to an *idealized* crypto oracle as achieved by prior works [RMS24, HPPY25]. Perfect privacy enables a simpler dual PIR construction. If a PIR is so perfectly private, there is no difference in how the dual PIR performs when Advice is run with the real query or a substitute query. Thus, we can simply have Hint output a random state st (which also determines the client's query randomness) leading to a 2^{-s} loss in correctness probability. However, we can still apply the impossibility of dual PIR when considering $k = \Omega(s)$. In particular, this construction never resamples crypto oracles (Recon replays the client with the real oracle O), so this is our one lower bound where Init may retain full crypto oracle access.

1.2.3 Symmetric PIR without Online Public-Key Cryptography

SPIR without Online Public-Key Cryptography. We present the first construction of SPIR that uses client-side preprocessing to avoid any public-key operations during online queries. Notably, our construction does not require per-client storage by the server. If per-client storage is allowed, then the problem is related to OT-extension [IKNP03]. We start from recent PIR with client preprocessing schemes [CK20, CHK22, RMS24, HPPY25], that preprocess a database by storing parities of small random sets that we refer to as hints. To add data privacy, we have the server one-time pad each hint during offline preprocessing without learning the hints (using generic 2PC protocol for example). At query time, the client sends the hint alongside the normal PIR query so the server can remove the one-time pad and return the correct answer. There are additional complications around data privacy as prior schemes use parallel repetition or require the server to return two XORs [RMS24, HPPY25]. We resolve these issues with a more complex hint structure including backup hints [CHK22] and the client randomly permuting hints to maintain query privacy.

Limits of SPIR with Preprocessing. We also prove stronger limits of SPIR with preprocessing beyond our broad PIR lower bounds. In the random oracle model, we prove that no perfectly private, single-server SPIR relative with s bits of client-side preprocessing can support batch queries of size $k > s$ regardless of online query communication costs, (even if the preprocessing is done "for free" without the adversary seeing it). Furthermore, we extend prior work [GIKM98] to prove that offline and online phases of SPIR supporting k queries require k-bits of correlated randomness. Our proofs utilize compression techniques as opposed to our more complex reductions using dual PIR.

2 Related works

We note that there is a long literature on single-server PIR without preprocessing (see [KO97, CMS99, Pai99, DJ01, IKOS04, GR05, OS07, ACLS18, GH19, PT20, MCR21, MW22, MR23, PSY23, BPSY24, BMW24] and references therein as examples). In this section, we mainly survey results concerning single-server PIR with preprocessing.

PIR with Server-Side Preprocessing. The model was first proposed by Beimel, Ishai and Malkin [BIM00] in the server-side preprocessing setting where the server can preprocess the database in hopes of faster online queries. There have been several works in this line including [BIPW17, CHR17, HOWW19, BHW19, PY22] leading to the breakthrough work of Lin, Mook and Wichs [LMW23] that obtained $\tilde{O}(1)$ online query communication and computation in the public-key DEPIR with server preprocessing model from the ring LWE assumption. Recent work also ruled out the existence of restricted classes of such schemes built from blackbox cryptography [LMW25]. We note that the lower bounds of [LMW25] apply to DEPIR schemes whose preprocessing may query the crypto oracle (under structural restrictions on the query protocol), whereas our DEPIR result (Section 4.5) allows arbitrary query protocols but requires the preprocessing to make no oracle queries, making the two results incomparable. Lastly, a recent work [CIMR25] presented a construction of secret-key DEPIR with $O(n^{\epsilon})$ communication from the LPN assumption with parameters that are not known to imply public-key cryptography.

PIR with Client-Side Preprocessing. Using client-side preprocessing to reduce the amount of server computation was first studied by Patel, Persiano, and Yeo [PPY18] that showed how to obtain a sublinear number of server public-key operations in a single-server PIR. The breakthrough

work of Corrigan-Gibbs and Kogan [CK20] showed that it is possible to even obtain sublinear total computation during query time. There has been a long line of follow-up works with improved constructions such as [KC21, SACM21, CHK22, MZRA22, LP23b, Yeo23a, GZS24, LP24]. By relying upon FHE during query time, it was shown that single-server schemes could obtain both $\tilde{O}(1)$ communication and sublinear computation [ZLTS23, LP23a].

More recent works starting from Piano [ZPZS24] along with follow-up works [RMS24, HPPY25, WR25] present single-server PIR with preprocessing schemes using only one-way functions with $t = \tilde{O}(n/s)$ communication with s-bit client storage. There also exist works that use heavy public-key cryptography during the offline phase to obtain sublinear communication while the query algorithm only uses one-way functions (see [CK20, CHK22, FLLP24]). However, these constructions only support $\tilde{O}(s)$ queries per offline phase when using s-bit client storage. If they wish to support more queries, they must re-execute their offline phase that requires public-key operations during queries (or streaming the entire database). Finally, we note that all prior single-server constructions (to our knowledge) may be modified to support an unbounded number of queries in the following way (this is effectively the approach used in [ZPZS24, RMS24, HPPY25, WR25] as well). Roughly speaking, in these constructions, a single offline phase supports $\tilde{O}(s)$ queries for s-bit client storage. Over the $\tilde{O}(s)$ queries, the client can stream the entire database and perform the offline phase locally to compute the next s-bit client storage. This avoids the usage of any public-key cryptography during online queries, but incurs $t = \Omega(n/s)$ amortized communication that is shown to be required by our lower bound. One can view our lower bound as showing that this approach is optimal in terms of communication complexity. Finally, prior works also consider information-theoretic security including [CK20, ISW24, SWZ24].

Lower Bounds. Di Crescenzo, Malkin, and Ostrovsky [DMO00] were the first to show that sublinear communication (i.e., non-trivial) single-server PIR without any preprocessing implies oblivious transfer. As oblivious transfer may be used to build key agreement, this may be combined with the well-known separation of Impagliazzo and Rudich [IR89] (and improvement by Barak and Mahmoody [BM09]), which shows that there is no key agreement between unbounded computational parties with polynomial queries to a random oracle. Furthermore, there are known server computation lower bounds for PIR without preprocessing. Beimel, Ishai and Malkin [BIM00] show that PIR (both single- and multi-server) require linear server computation.

For single-server PIR, Dujmovic and Hajiabadi [DH24] show that a linear number of public-key operations is necessary without preprocessing. In the preprocessing model, their lower bound shows that the server's total cryptographic operations (across both the offline and online phases) must be $\Omega(n)$. Their result makes no claims about, for example, the online server cryptographic operations for PIR with preprocessing in which the server makes $\Omega(n)$ public-key operations. In contrast, our lower bound is stronger as it shows that $\Omega(n/s)$ amortized cryptographic bit operations must be performed by the server during online queries.

For the preprocessing model, there are known lower bounds on the trade-offs between client storage and query time in both the client [CK20, CHK22, Yeo23b] and server [BIM00, PY22] preprocessing models. However, all of these lower bounds only apply to the restricted class of constructions with very strong non-encoding server requirements such that the server must use the n-bit database without modification to serve PIR queries. In contrast, our computation lower bounds apply to schemes in which the server may arbitrarily encode the database for use in answering queries, so long as the encoding is computed without crypto oracle queries. Furthermore, none of the techniques in this line of work yield any communication lower bounds.

Ishai, Shi, and Wichs [ISW24] were the first to present communication lower bounds for single-server PIR with client preprocessing. First, they showed that there exists no information-theoretic construction with s-bit client storage and query communication $t = O(n/s)$. In particular, such a construction implies the existence of a one-way function. For the restricted class of database-oblivious and single roundtrip query constructions, they showed that a s-bit client scheme with $t = o(n/s)$ query communication would imply a separation between complexity classes SZK and BPP that remains an open problem in complexity theory in many natural settings (such as relative to a random oracle). Our new result cannot be directly compared to [ISW24]. Notably, we show a stronger (unconditional) lower bound on the use of blackbox cryptography for a class of PIR that is broader in several respects (multi-round and database-dependent queries). However, our result does not have the same complexity theoretic implications. In particular, if one were to show that another assumption X (e.g., learning parity with noise, attribute-based encryption, etc.) or a white-box reduction from one-way functions implied a too-efficient PIR with preprocessing, then a corollary of the prior work [ISW24] would be that X implies a hard problem in SZK. This is something which our result does not comment on. Additionally, both the barrier in [ISW24] and our communication lower bounds only apply to restricted classes of schemes. We note the imposed restrictions are incomparable. The barrier in [ISW24] required the query algorithm to be single roundtrip and ensure that all client requests to be database-oblivious (i.e., client requests may only depend on the queried index and internal randomness). In contrast, our work makes no restrictions on the communication (the query algorithm may have multiple rounds and depend on the database). Instead, we require one of the following: the server to make $o(n/s)$ cryptographic operations, all server cryptographic operations depend only on the communication transcript or obtain perfect privacy in the idealized model. In the first two cases, we additionally require the server's one-time database encoding to be computed without cryptographic operations. The perfect-privacy case does not need this restriction and instead applies to schemes deriving their query randomness from the random oracle (see Section 5.3).

Symmetric PIR. Symmetric PIR (SPIR), where the contents of the database are also protected, has been studied for decades (see [GIKM98, FIPR05, ALP+21] and references therein). There are ways to generically construct SPIR from (non-symmetric) PIR using, for example, oblivious PRFs [FIPR05, ALP+21], which works in the preprocessing setting for certain constructions. To our knowledge, these generic transformations require public-key operations at query time. One could also try to use these public operations to the offline phase (such as using OT-extension [IKNP03]), but this would require the server to store client-specific state that is undesirable as the server storage would grow with the user base.

Notice that for SPIR (unlike PIR), we *must* have public-key operations in the offline or online settings, since it is equivalent to oblivious transfer. This means we cannot use tricks like streaming the database to the client in the offline phase that were used in prior PIR works. We discuss this further in Section 6.

3 Preliminaries

Notation. Throughout our paper, we use n to denote the size of a database D, which we consider to be a bitstring $D \in \{0,1\}^n$. We additionally take λ to be an integer which represents the security parameter of the schemes in question. We also make use of the notation $[n] = \{1, \ldots, n\}$. For a tuple of indices/queries $Q = (q_1, \ldots, q_k) \in [n]^k$, we use $D[Q]$ to denote $(D[q_1], \ldots, D[q_k])$.

In this work, each algorithm denotes a possibly randomized (Turing machine) computation, which takes zero or more inputs and has one or more outputs. A (two-party) protocol is a pair of interactive computers each with their own zero or more inputs. The messages passed between the computers define the transcript. At the end of the protocol, the machines output their results. We denote a protocol Π with inputs x_i, outputs y_i, and transcripts τ, as $\Pi(x_1; x_2) \to (y_1; y_2 \mid \tau)$. If one party has no output, we denote its output as $\perp$. Sometimes, we ignore the transcript τ when it is not relevant. We say an algorithm or protocol is *efficient* if there exists a polynomial p, such that for any input(s) x, the runtime of algorithm or protocol is at most $p(|x|)$.

In our pseudocode, we use $x \leftarrow y$ to denote assigning x the value of y. Likewise we use $x \xleftarrow{\$} y$ to denote sampling (uniformly) from a set y and assigning the value to x (e.g. $x \xleftarrow{\$} [n]$ denotes selecting a random integer between 1 and n inclusive). The item y may also be a randomized function, which could just be viewed as running the function with fresh randomness. We use $\mathsf{negl}(\cdot)$ as a placeholder for some negligible function, i.e., one where for every polynomial p there is some n such that, $\mathsf{negl}(n') < 1/p(n')$, for every $n' \geq n$. For conceptual clarity, we use **bold** font to denote random variables when measuring probabilities or other information-theoretic quantities. For example, we may have a variable $D \in \{0,1\}^n$ which represents a database, which corresponds to $\mathbf{D}$ to represent a uniformly random choice of database.

Information theory. At various points in our results, we are concerned with the minimum entropy or min-entropy that we define in Definition 3.1. This quantity essentially measures the predictability of a random variable. In particular, if a random variable $\mathbf{X}$ has min-entropy at least h, then an input-less algorithm can predict the value of $\mathbf{X}$ only with probability at most 2^{-h}. In particular, for all $\mathcal{A}$, $\Pr_{\mathbf{X}}[\mathcal{A} = \mathbf{X}] \leq 2^{-H_\infty(\mathbf{X})}$.

In our results, it is also useful for us to consider a slightly more general version of min-entropy as well, called conditional min-entropy. Although there are few ways to define this quantity, our choice is related to the predictability of a random variable $\mathbf{X}$, given some side information (also a random variable) $\mathbf{Y}$, which may be correlated with $\mathbf{X}$. Using our definition however, it is easy to see that for any algorithm $\mathcal{A}$, $\Pr_{\mathbf{X},\mathbf{Y}}[\mathcal{A}(\mathbf{Y}) = \mathbf{X}] \leq 2^{-H_\infty(\mathbf{X}|\mathbf{Y})}$.

Definition 3.1. *Let $\mathbf{X}, \mathbf{Y}$ be random variables supported on $\mathcal{X}$ and $\mathcal{Y}$ respectively. The* min-entropy *of $\mathbf{X}$ is defined as*

$$H_\infty(\mathbf{X}) := -\log\left(\max_{x \in \mathcal{X}} \Pr_{\mathbf{X}}[\mathbf{X} = x]\right).$$

We further define the conditional min-entropy *of $\mathbf{X}$ given $\mathbf{Y}$ as*

$$H_\infty(\mathbf{X} \mid \mathbf{Y}) := -\log\left(\mathbb{E}_{y \sim \mathbf{Y}}\left[\max_{x \in \mathcal{X}} \Pr_{\mathbf{X}}[\mathbf{X} = x \mid \mathbf{Y} = y]\right]\right).$$

We use the following as a supporting lemma in the proof of our main result. This lemma provides a concrete bound on the amount that min-entropy can drop when given some additional side-information.

Lemma 3.2 ([DRS04])**.** *If $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ are random variables and $\mathbf{Y}$ is supported on at most k elements, then*

$$H_\infty(\mathbf{X} \mid \mathbf{Y}, \mathbf{Z}) \geq H_\infty(\mathbf{X}\mathbf{Y} \mid \mathbf{Z}) - \log k \geq H_\infty(\mathbf{X} \mid \mathbf{Z}) - \log k.$$

In addition to min-entropy, we denote the statistical distance between two random variables $\mathbf{X}$ and $\mathbf{Y}$ with the same support $\mathcal{U}$ as $\Delta(\mathbf{X}, \mathbf{Y}) := \frac{1}{2}\sum_{u \in \mathcal{U}} |\Pr[\mathbf{X} = u] - \Pr[\mathbf{Y} = u]|$. We also use the fact that the best unbounded distinguisher between $\mathbf{X}$ and $\mathbf{Y}$ successfully identifies the correct random variable with probability $1/2 + \Delta(\mathbf{X}, \mathbf{Y})/2$.

3.1 Crypto Oracles

This work considers relativized worlds where some or all parties have access to (possibly randomized) oracles. In particular, we denote an algorithm $\mathcal{A}$ with access to an oracle O as $\mathcal{A}^{\mathsf{O}}$. In relevant settings, the presence of a random oracle requires us to change the sample space of a probability experiment to additionally consider the underlying randomness of the random oracle O (for example, our definitions of PIR with preprocessing in Definition 3.7).

We will utilize the notion of *crypto oracles*[1] that encapsulates the usage of powerful blackbox cryptography primitives. We use the definition introduced by Lin, Mook, and Wichs [LMW25] that considers stateless simulators, but first considered by Dujmovic and Hajibadi [DH24].[2]

Definition 3.3 (Due to [LMW25])**.** *A crypto oracle is a function $\mathcal{B}^R$ where B is a stateless, polynomial time, deterministic Turing machine with oracle access to a secret random function $R : \{0,1\}^* \to \{0,1\}$.*

In the same work [LMW25], it was shown that crypto oracles may be used to implement both virtual blackbox obfuscation and the generic multilinear group model. We extend this result to (unsurprisingly) show that one can implement public-key fully homomorphic encryption (with key and ciphertext length independent of the size of the circuit evaluated), which can be used to construct PIR directly. We refer the reader to [KL14] for security definitions relevant for public-key encryption.

Lemma 3.4. *There exists a crypto oracle $\mathcal{B}^R$ that implements public-key fully homomorphic encryption with $\mathsf{poly}(\lambda)$-size keys.*

Proof. We construct a crypto oracle $\mathcal{B}^R$ providing interfaces $(\mathsf{KG}^R, \mathsf{Enc}^R, \mathsf{Eval}^R, \mathsf{Dec}^R)$, where $R : \{0,1\}^* \to \{0,1\}^\lambda$ is the underlying secret random function.[3]

Since $\mathcal{B}^R$ must be deterministic, the randomized algorithms KG and Enc take their random coins $\rho \in \{0,1\}^\lambda$ as an explicit input, and all indices are derived using R with the distinct prefixes kg, enc, eval, and pad.

- $\mathsf{KG}^R(1^\lambda; \rho)$: Set $sk = \rho$, set $pk = R(\mathsf{kg} \| sk)$ and output (pk, sk).
- $\mathsf{Enc}^R(pk, m; \rho)$: Compute the index $i = R(\mathsf{enc} \| pk \| \rho)$ and output $\mathsf{ct} = (i, R(\mathsf{pad} \| pk \| i) \oplus m)$.
- $\mathsf{Eval}^R(pk, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$: Parse each $\mathsf{ct}_j = (i_j, c_j)$, recover $m_j = c_j \oplus R(\mathsf{pad} \| pk \| i_j)$. Then, compute $m^* = f(m_1, \ldots, m_\ell)$ and the index $i^* = R(\mathsf{eval} \| pk \| f \| \mathsf{ct}_1 \| \cdots \| \mathsf{ct}_\ell)$, and output $(i^*, R(\mathsf{pad} \| pk \| i^*) \oplus m^*)$.
- $\mathsf{Dec}^R(sk, \mathsf{ct})$: Recover $pk = R(\mathsf{kg} \| sk)$, parse $\mathsf{ct} = (i, c)$, and output $c \oplus R(\mathsf{pad} \| pk \| i)$.

Both pk and sk have size $\mathsf{poly}(\lambda)$. Standard correctness is immediate. Encryption is public-key: Enc^R requires only pk and makes no use of sk. Note that IND-CPA security holds against any adversary making $q = \mathrm{poly}(\lambda)$ queries to $\mathcal{B}^R$. The challenge ciphertext is $(i^*, R(\mathsf{pad} \| pk \| i^*) \oplus m_b)$ for $i^* = R(\mathsf{enc} \| pk \| \rho^*)$ with uniformly random coins ρ^*. The pad $R(\mathsf{pad} \| pk \| i^*)$ is revealed to

[1] In Appendix B, we discuss how our results generalize beyond crypto oracles.

[2] Note, the definition in [DH24] considers stateful simulators, but there does not seem to be any meaningful cryptographic primitive not captured by stateless simulators as pointed out in [LMW25].

[3] We use notation with λ-bit outputs to simplify our presentation. This can be implemented without loss of generality by querying R with distinct prefixes.

the adversary only through a ciphertext under the same index i^* whose plaintext the adversary knows. This requires an Enc^R or Eval^R query whose derived index equals i^*, or a Dec^R query with a secret key sk' such that $R(\mathsf{kg}\|sk') = pk$. Since R is a random function evaluated on distinct inputs (the prefixes ensure that index derivations never coincide with pad evaluations) and ρ^* is hidden, each of these happens with probability at most $q/2^\lambda$. Otherwise, the pad is uniformly random and independent of the adversary's entire view, so the challenge ciphertext is uniformly random regardless of b, and the adversary gains no advantage. Homomorphic correctness follows because Eval^R decrypts and re-encrypts internally using R, without revealing any plaintext to the caller. □

Theorem 3.5 (Due to [LMW25] and Lemma 3.4)**.** *There exists crypto oracles that implement the following:*

- *(Trivially) random oracles,*
- *Virtual blackbox obfuscation for Turing machines,*
- *Generic multilinear groups.*
- *Public-key fully homomorphic encryption.*

The above is very powerful as it enables instantiating a wide range of powerful cryptographic primitives in a blackbox manner including one-way functions, fully homomorphic encryption and obfuscation, which will be relevant as they have been used in past constructions of PIR with preprocessing.

We note that a single crypto oracle may be used to implement arbitrarily many different crypto oracles that implement different cryptographic primitives and/or ideal models. In particular, a single random function R may be used to derive an arbitrary number of independent random functions (for example, by prepending with distinct prefixes). Therefore, without loss of generality, we can consider a single crypto oracle that implements multiple (or all) of the above cryptographic primitives and idealized models.

We will need the following result that shows it is possible to sample a random crypto oracle such that a polynomial number of queries agree with a previously sampled crypto oracle using a random oracle RO. In particular, we consider a crypto oracle O along with a polynomial set of queries S. We want to sample another random crypto oracle O' such that $\mathsf{O}(q) = \mathsf{O}'(q)$ for all $q \in S$. For convenience, we will denote the set of queries and results as $\mathsf{O}(S) = \{(q, \mathsf{O}(q)) \mid q \in S\}$. The following Lemma 3.6 is inspired by a step at the heart of blackbox separations in the random oracle model [IR89, BM09]. After learning the oracle queries relevant to a transcript, an attacker samples a fresh random oracle that agrees with them and simulates the honest parties. For a random oracle this resampling is trivial, whereas we show it remains possible for crypto oracles, where the resampled oracle must again be of the form $\mathcal{B}^{R'}$.

Lemma 3.6. *Let O be a crypto oracle and let S be a set of queries for O of size $|S| = \mathrm{poly}(\lambda)$. For any $\zeta = \mathrm{poly}(\lambda)$, there exists a deterministic algorithm $\mathcal{A}$ with access to a random oracle RO such that $\mathcal{A}^{\mathsf{RO}}(\mathsf{O}(S))$ outputs a new uniformly random crypto oracle O' with $\mathsf{O}(q) = \mathsf{O}'(q)$ for all $q \in S$, except with probability $e^{-\zeta}$ over the random choice of RO. Moreover, conditioned on the algorithm succeeding, the output O' is distributed exactly as a uniformly random crypto oracle conditioned on the event that $\mathsf{O}'(q) = \mathsf{O}(q)$ for all $q \in S$.*

Proof. As O is a crypto oracle, there exists some stateless, deterministic simulator $\mathcal{B}^R$ that makes at most $t = \mathrm{poly}(\lambda)$ queries to the private random function R. Across all oracle queries $\mathsf{O}(q)$

for all $q \in S$, we note that $\mathcal{B}^R$ makes at most $|S| \cdot t$ queries to R. First, we use RO to generate $k = 2^{|S|\cdot t} \cdot \zeta$ different random $\mathsf{RO}_1, \ldots, \mathsf{RO}_k$ by setting $\mathsf{RO}_i(\cdot) = \mathsf{RO}(i \mid\mid \cdot)$ where i is represented using a $\lceil \log_2 k \rceil$-bit string. Note that $\lceil \log_2 k \rceil = O(|S| \cdot t + \mathrm{poly}(\lambda)) = \mathrm{poly}(\lambda)$ since both $|S|$ and t are polynomial-sized, so all inputs to RO are poly-sized. Afterwards, our algorithm constructs O_i by executing $\mathcal{B}^{\mathsf{RO}_i}$ and checks whether $\mathsf{O}_i(q) = \mathsf{O}(q)$ for all $q \in S$. The final output oracle O' is O_i for the first index i satisfying the above condition. Now, we bound the probability that the algorithm finds such an O_i. First, we note that the probability that any O_i constructed using RO_i will match O at all queries in S is at least $2^{-|S|\cdot t}$, which is the probability that RO_i exactly matches the underlying private random function R used by the simulator $\mathcal{B}^R$ corresponding to O. Note, there might exist more choices of random functions that result in oracles that agree on queries in S, but that would only increase this probability. Therefore, the probability that all $k = 2^{|S|\cdot t} \cdot \zeta$ random oracles fail to produce a crypto oracle that matches the original oracle O on all queries in S is at most $(1 - 2^{-|S|\cdot t})^k \leq e^{-\zeta}$ using the fact that $(1 - 2^{-|S|\cdot t})^{2^{|S|\cdot t}} \leq e^{-1}$. Finally, we argue the distribution of the output conditioned on success. Each candidate oracle $\mathsf{O}_i = \mathcal{B}^{\mathsf{RO}_i}$ for an independent uniformly random function RO_i is an independent sample of a uniformly random crypto oracle, and the algorithm outputs the first candidate agreeing with O on all of S. By the standard rejection sampling argument, the first accepted sample among independent and identically distributed candidates, conditioned on acceptance occurring, is distributed as a single candidate conditioned on acceptance; that is, as a uniformly random crypto oracle conditioned on agreeing with $\mathsf{O}(S)$. □

Note, as the above algorithm $\mathcal{A}$ is deterministic, it will output the same newly sampled oracle O' when given access to the same random oracle RO as well as input $\mathsf{O}(S)$.

3.2 PIR with Preprocessing

We focus on single-server PIR in the client preprocessing model, a notion that we formally present in Definition 3.7. In this model, the client is allowed to preprocess the database $D \in \{0,1\}^n$ during the offline phase; then, in the online phase, the client issues its set of queries Q. In our definition, we choose a slightly weaker model that enables constructions with worse privacy and weaker functionality. As we prove lower bounds, this makes our results stronger as they will apply to a wider range of schemes. We remind the reader that the syntax $D[Q]$ for a tuple $Q = (q_1, \ldots, q_k)$ is used to denote the tuple $(D[q_1], \ldots, D[q_k])$.

Definition 3.7 (Private Information Retrieval)**.** *A k-batch* private information retrieval with preprocessing scheme (PIR) *is a triplet of efficient two-party protocols with access to an oracle* O, $\Pi^{\mathsf{O}} = (\mathsf{Init}^{\mathsf{O}}, \mathsf{Setup}^{\mathsf{O}}, \mathsf{Query}^{\mathsf{O}})$*, parameterized by a database size n, batch size k, total communication t, and state size s with the following syntax:*[4]

- $\mathsf{Init}^{\mathsf{O}}(1^\lambda, D) \to E$*, a (possibly randomized) algorithm run by the server to encode the original database* $D \in \{0,1\}^n$ *into some encoded database E. At the end of the protocol, the client receives no output and the server receives an encoded database E.*
- $\mathsf{Setup}^{\mathsf{O}}(1^\lambda; E) \to (\mathsf{st}; \perp)$*, where the client receives as input the security parameter* 1^λ *and the server receives the encoded database E. At the end of the protocol, the client receives a state* $\mathsf{st} \in \{0,1\}^s$ *and the server receives no output.*

[4]Note that k, t, s are typically functions of λ and n that we omit for simplicity.

– $\mathsf{Query}^{\mathsf{O}}(\mathsf{st}, Q; E) \to (\mathsf{ans}; \perp \mid \tau)$, *where the client takes its precomputed state* st *and an ordered set of queries* $Q \in [n]^k$ *and the server receives the encoded database* E*. At the end of the protocol, the client receives* $\mathsf{ans} \in \{0,1\}^k$ *and the server receives no output. We denote the messages sent between the client and server as* $\tau \in \{0,1\}^t$*. Where relevant, we will also separate* $\tau = (\tau_{\mathsf{Client}}, \tau_{\mathsf{Server}})$ *to distinguish the messages* τ_{Client} *sent by the client to the server and the messages* τ_{Server} *sent by the server to the client.*

A PIR is ε-correct *if for all sufficiently large* λ*,* $Q \in [n]^k$*, and* $D \in \{0,1\}^n$*,*

$$\Pr\left[\mathsf{Query}^{\mathsf{O}}(\mathbf{st}, Q; \mathbf{E}) = (D[Q]; \perp) \,\middle|\, \substack{\mathbf{E} \xleftarrow{\$} \mathsf{Init}^{\mathsf{O}}(1^\lambda, D) \\ (\mathbf{st}; \perp) \xleftarrow{\$} \mathsf{Setup}^{\mathsf{O}}(1^\lambda; \mathbf{E})}\right] \geq \varepsilon.$$

A PIR is δ-private *if for all sufficiently large* λ*, tuples* $Q_0, Q_1 \in [n]^k$*,* $D \in \{0,1\}^n$*, and adversaries* $\mathcal{A}$ *issuing* $\mathrm{poly}(\lambda)$ *queries to* O*,*

$$\Pr\left[\mathcal{A}^{\mathsf{O}}(1^\lambda, \boldsymbol{\tau}, \mathbf{E}) = b \,\middle|\, \substack{\mathbf{E} \xleftarrow{\$} \mathsf{Init}^{\mathsf{O}}(1^\lambda, D) \\ (\mathbf{st}; \perp) \xleftarrow{\$} \mathsf{Setup}^{\mathsf{O}}(1^\lambda; \mathbf{E}) \\ b \xleftarrow{\$} \{0,1\} \\ (\mathbf{ans}; \perp \mid \boldsymbol{\tau}) \xleftarrow{\$} \mathsf{Query}^{\mathsf{O}}(\mathbf{st}, \mathbf{Q}_b; \mathbf{E})}\right] \leq \frac{1}{2} + \delta.$$

When $\delta = 0$ *we say the PIR is* perfectly private*.*

We note that the above definition is flexible enough to cover almost all prior studied PIR definitions. For PIR with client preprocessing, we essentially allow the server to perform arbitrary encoding of the database that may be used by the client to generate their private state.

While our definition places no restriction on the encoding algorithm, our computation lower bounds will apply to the class of schemes whose encoding is computed without access to the crypto oracle, which we formalize as follows.

Definition 3.8 (Oracle-Free Encoding)**.** *A PIR* $\Pi^{\mathsf{O}} = (\mathsf{Init}^{\mathsf{O}}, \mathsf{Setup}^{\mathsf{O}}, \mathsf{Query}^{\mathsf{O}})$ *has an* oracle-free encoding *if the encoding algorithm* Init *makes no queries to the oracle* O*. In this case, we drop the oracle superscript and simply write* $\mathsf{Init}(1^\lambda, D) \to E$*.*

Our computation lower bounds and our first two communication lower bounds apply to schemes with oracle-free encodings, while Setup and Query may use the oracle O arbitrarily. To our knowledge, this captures every existing single-server PIR with client preprocessing construction most of which have the server store the database unmodified and all cryptographic work occurs inside the interactive Setup and Query protocols, including FHE-based offline phases [CK20, CHK22, FLLP24]. Our definition also encompasses public-key (and unkeyed) doubly efficient PIR (DEPIR) [BIPW17, LMW23]: the server runs, inside Init, a key-generation step together with the database encoding, and outputs both the public key and the encoded database as E (discarding the secret key), while the Setup algorithm is essentially ignored as the client receives no state (see Section 4.5).

On the other hand, our main lower bounds do not apply to schemes whose database encoding queries the crypto oracle (see Remark 4.10 for plausible relaxations). In particular, our lower bound aligns with the server's database encoding for the state-of-the-art public-key DEPIR [LMW23] but not earlier schemes that use cryptographic preprocessing for the server's database encoding [BIPW17]. Although, we note none of these schemes make blackbox usage of cryptography and, thus, are outside the scope of our lower bounds. Similarly, our definition itself does

not capture secret-key DEPIR, where the server's encoded database may depend on the client's privately generated state [BIPW17, CHR17, LMW25, CIMR25], as our proofs require the server's encoding to be performed *first* before generating the client's private state.

Offline Phase. One can note that our definition is *essentially agnostic* to Setup. In particular, the adversary is not given the transcript for the offline phase when Setup is executed. As a result, our lower bounds will apply to all choices of Setup including the two popular choices of relying on fully-homomorphic encryption [CK20, CHK22, FLLP24] and streaming the entire database [ZPZS24, RMS24, HPPY25]. In fact, our lower bound applies even if we assume a setting where the offline phase is run securely and the adversary only appears during the online query portion.

Batch Queries. Our definition considers the setting of batch PIR where it is assumed that all k queried entries are given at once. Again, this only strengthens our lower bound as it applies to both batch and standard, single-query PIR. In particular, it is easy to see that it is straightforward to construct a k-batch query PIR from a single-query PIR by performing k queries in sequence.

Unrestricted Query Algorithm. Additionally, we note that our definition does not make any restrictions on the query algorithm of the PIR with preprocessing scheme. In contrast, we note that the prior work [ISW24] considers a restricted class of query algorithms that are database-oblivious (client requests are entirely independent of the database) and single roundtrip. Our lower bounds apply to schemes beyond these restrictions including those that are multiple roundtrips and may depend on the contents of the database.

3.3 Subkey prediction

We will rely upon a result about subkey prediction due to Bellare, Kane, and Rogaway [BKR16]. Subkey prediction considers the ability of an adversary to predict a uniformly random subkey of a large key, given some short leakage (side information). In particular, the large key consists of n independent and uniformly random bits and the adversary is interested in a subkey consisting of exactly k uniformly random locations in the large key. Additionally, the adversary obtains t bits of arbitrary leakage (side information) of the large key, chosen independently of the subkey's uniformly random locations. For example, one can imagine the adversary is able to preprocess the large key of n bits into any arbitrary t bits of information before the indices of the subkey are chosen. Afterwards, the k uniformly random locations of the subkey are provided and the adversary must predict the k bits of the subkey using the t bits of leakage.

In more detail, suppose a random n-bit key $\mathbf{D} \in \{0,1\}^n$ is chosen. The adversary is first able to output t-bit leakage denoted $\mathcal{A}_1(\mathbf{D})$. Afterwards, k random locations are chosen $\mathbf{Q} \in [n]^k$. Lastly, the adversary is given the t-bit leakage and k random locations and outputs its guess denoted by $\mathcal{A}_2(\mathbf{Q}, \mathcal{A}_1(\mathbf{D}))$. We use the following lemma directly from [BKR16]. Before we present the lemma, we recall some notation that will be important. We denote by $B_n(r)$ the radius r ball centered at the origin of the n-dimensional boolean hypercube; that is, all points of $\{0,1\}^n$ whose distance is at most r from the all-zero vector. Then, we see that $|B_n(r)| = \sum_{i=0}^{r} \binom{n}{i}$. Additionally, we denote the radius $\mathsf{rd}_n(N) := \max\{\hat{r} \in \mathbb{Z}_{\geq 0} : |B_n(\hat{r})| \leq N\}$ as the radius of the largest ball centered at the all-zero vector of the boolean hypercube that contains at most N points.

Lemma 3.9 (Subkey Prediction [BKR16])**.** *For sufficiently large integers n, t, k and for all (computation-*

ally unbounded) adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr[\mathcal{A}_2(\mathbf{Q}, \mathcal{A}_1(\mathbf{D})) = \mathbf{D}[\mathbf{Q}]] \leq \frac{1}{2^{n-t}} \cdot \sum_{i=0}^{\min(1+\mathsf{rd}_n(2^{n-t}),n)} \binom{n}{i} \cdot \left(1 - \frac{i}{n}\right)^k$$

where $\mathbf{Q} \xleftarrow{\$} [n]^k$ *are the k subkey locations,* $\mathbf{D} \xleftarrow{\$} \{0,1\}^n$ *is the n-bit large key, adversary* $\mathcal{A}_1$ *outputs at most t bits of leakage and* $\mathcal{A}_2$ *outputs k bits.*

3.4 Blackbox constructions

In our work, we will consider blackbox constructions. We borrow the following definitions directly from the work of Reingold, Trevisan and Vadhan [RTV04].

Definition 3.10 (Blackbox Constructions)**.** *Let* $\mathcal{P}$ *and* $\mathcal{Q}$ *be two primitives. A blackbox construction of primitive* $\mathcal{Q}$ *from primitive* $\mathcal{P}$ *consists of two PPT algorithms* (Q, S) *satisfying the following:*

- *For oracle P that implements primitive* $\mathcal{P}$*, the algorithm* Q^P *implements primitive* $\mathcal{Q}$*.*
- *For any oracle P implementing primitive* $\mathcal{P}$ *along with any computationally unbounded oracle adversary A successfully breaking the security of* Q^P*, then the oracle adversary* $S^{P,A}$ *breaks the security of P.*

Next, we will say that a primitive exists relative to an oracle using the following definition.

Definition 3.11. *A primitive* $\mathcal{P}$ *exists relative to an idealized oracle* $\mathcal{O}$ *if there exists an implementation P for primitive* $\mathcal{P}$ *that is computable by a PPT algorithm with access to idealized oracle* $\mathcal{O}$*. Furthermore, there exists no (computationally unbounded) adversary that makes at most polynomial number of queries to idealized oracle* $\mathcal{O}$ *that breaks the security of the implementation P.*

Finally, we will say that a blackbox construction relativizes using the following result from [RTV04].

Theorem 3.12 (Relativization)**.** *If there is a blackbox construction of primitive* $\mathcal{Q}$ *from primitive* $\mathcal{P}$*, then the following two statements holds:*

- *If primitive* $\mathcal{P}$ *exists relative to idealized oracle* $\mathcal{O}$*, then primitive* $\mathcal{Q}$ *also exists relative to idealized oracle* $\mathcal{O}$*.*
- *If there exists a blackbox construction of primitive* $\mathcal{P}$ *from idealized oracle* $\mathcal{O}$*, then there also exists a blackbox construction of primitive* $\mathcal{Q}$ *from idealized oracle* $\mathcal{O}$*.*

Note, we can use the above to essentially show the following. If we prove that any cryptographic primitive cannot exist relative a crypto oracle O, then we prove that there cannot exist any blackbox construction of the cryptographic primitive using any primitive that can be built from a crypto oracle O (including random oracles, generic multilinear groups and virtual blackbox obfuscation as shown in Theorem 3.5 taken from [LMW25]).

In the above definitions, we consider fully blackbox constructions from [RTV04]. One can also consider semi-blackbox constructions where the security reduction is not necessarily blackbox and may make use of the internal structure of the adversary. In this paper, we consider fully blackbox constructions for simplicity, but our results may be extended to rule out semi-blackbox constructions using the same techniques explained in [LMW25]. In particular, most cryptographic primitives such as encryption, obfuscation, and oblivious transfer enable the embedding property introduced in [RTV04] which extends our results to semi-blackbox constructions for those primitives.

4 Computation Lower Bounds

We present our main result: an amortized computation lower bound of $\Omega(n/s)$ for any single-server PIR with client preprocessing with s-bit client storage supporting $k = \Theta(s)$ queries.

Theorem 4.1. *Relative to any crypto oracle* O,[5] *there does not exist a single-server PIR with preprocessing scheme with an oracle-free encoding (Definition 3.8) that is ε-correct and δ-private with s-bit client storage and total computation of $t = o(n)$ supporting $k = \Omega(s)$ queries with $k \leq 0.19n$ against computationally unbounded adversaries making $\mathrm{poly}(\lambda)$ queries to O for any choice of $s = \omega(\log(n + \lambda))$, $\varepsilon = \Omega(2^{-s})$ and $\delta \leq \varepsilon/3$. In more detail, there exists no such scheme satisfying both of the following efficiency measures:*

- *Total communication of $o(n)$ bits.*
- *Total of $o(n)$ input bits to the crypto oracle O by the server during the online query protocol.*

Our above result unconditionally rules out the existence of any such efficient single-server PIR with preprocessing schemes with oracle-free encodings that make blackbox usage of any cryptography that may be implemented using a crypto oracle (see Section 3.4 for more details on blackbox constructions). This includes random oracles (one-way functions), virtual blackbox obfuscation, fully homomorphic encryption and idealized models such as generic multilinear groups (see Section 3.1). As discussed in Section 3.2, the oracle-free encoding restriction is satisfied by every existing single-server PIR with client preprocessing construction that we are aware of, most of which do not encode the database at all along with the state-of-the-art public-key DEPIR.

We note our lower bound only applies to schemes that support sufficiently large $k = \Omega(s)$ queries. This turns out to be a necessary restriction as there exist prior constructions that support a smaller number of queries with $o(n/s)$ communication per query (see Appendix A).

Finally, we remark on the wide applicability of our lower bound towards the parameterizations of ε and δ. In particular, our lower bound rules out any scheme that is correct with exponentially small probability in s. Note, the requirement that $\delta \leq \varepsilon/3$ is essentially required as if $\delta > \varepsilon$ there are trivial algorithms that essentially do plaintext retrievals with probability δ (sacrificing privacy), but obtaining sufficient correctness. For more larger choices of correctness guarantees such as $\varepsilon = O(1)$ or $\varepsilon = 1/\mathrm{poly}(\lambda, n)$, we note our lower bound applies to PIR with preprocessing schemes with weaker privacy guarantees with non-negligible choices of δ.

Proof Overview. Our proof proceeds in three steps as follows. Each of the following sections is devoted to one step of the proof.

1. First, we present and define the new notion of dual PIR where the roles of the online queries and offline preprocessing in PIR with preprocessing are essentially switched.

2. Next, we present a reduction that any PIR with preprocessing that is secure relative to a crypto oracle O implies that there is a dual PIR relative to that same crypto oracle O such that the dual PIR's efficiency guarantees directly relate to the client storage size, communication and server cryptographic operations in the original PIR with preprocessing scheme.

3. Finally, we prove information-theoretically (and therefore relative to any O) that no dual PIR can be both correct and efficient, by reducing to the subkey prediction problem from leakage-resilient cryptography.

[5] In Appendix B, we explain how our results extend to *any* "consistently sampleable" oracles, of which crypto oracles are a subset.

4.1 Dual Private Information Retrieval

We introduce a notion that we call *dual* private information retrieval. At a high level, we reverse the order of the offline preprocessing and online querying that appears in the PIR with preprocessing setting. Indeed, for dual PIR, there is first some t-bit *advice* that must be generated without any knowledge (independently) of future queries by the client. Afterwards, the client is given a batch of k query indices $Q \in [n]^k$ to retrieve from the database $D \in \{0,1\}^n$. To answer their queries, the client may additionally request an arbitrary s-bit *hint* about the database D. After receiving this information though, the client must be able to correctly retrieve the queried indices, $D[Q] \in \{0,1\}^k$, using the t-bit advice and the s-bit hint.

The t-bit advice produced by Advice plays the role of the transcript of the online queries, whereas the s-bit hint computed using Hint is equivalent to the s-bit client storage from offline preprocessing in standard PIR with preprocessing. Furthermore, the efficiency guarantees have been swapped in dual PIR where the client obtains t bits in the first phase and s bits in the second phase. In contrast, a PIR with preprocessing scheme obtains an s bits of client storage in the offline phase and t bits of communication during online queries. For this reason we shall sometimes call the advice *the transcript* and say that the hint is the *secret*.

The definition of dual PIR that we present does not explicitly provide any privacy guarantees. It turns out that privacy is essentially baked into the syntax of the algorithms. In the PIR with preprocessing model, the privacy guarantees required by the query transcript viewed by the adversary during the online phase should not reveal information about the queried indices Q. In dual PIR, this privacy is essentially guaranteed since the client must generate the t-bit transcript without any knowledge of the future query indices. Similarly, we allow the client to compute Hint outside of the adversary's view similar to our definition of Setup in PIR with preprocessing (see Definition 3.7).

We formally present the definition of dual PIR below:

Definition 4.2 (Dual private information retrieval)**.** *A* dual private information retrieval *for database size n, batch size k, advice size t, and hint size s, shortened to a (n,k,t,s)-dual PIR, is a tuple of algorithms $\widetilde{\Pi} = (\mathsf{Advice}, \mathsf{Hint}, \mathsf{Recon})$ which satisfies the following syntax:*

- $\mathsf{Advice}(D) \to A$*, takes as input a database $D \in \{0,1\}^n$ and outputs a string of advice $A \in \{0,1\}^t$.*
- $\mathsf{Hint}(D, Q, A) \to H$*, takes as input a database $D \in \{0,1\}^n$, a batch of queries $Q \in [n]^k$ as well as advice $A \in \{0,1\}^t$ and outputs a hint string $H \in \{0,1\}^s$.*
- $\mathsf{Recon}(Q, A, H) \to \mathsf{ans}$*, takes as input queries $Q \in [n]^k$, advice $A \in \{0,1\}^t$ as well as hint string $H \in \{0,1\}^s$ and outputs the answer $\mathsf{ans} \in \{0,1\}^k$.*

*We say that a (n,k,t,s)-*dual PIR *is ε-correct if*

$$\Pr\left[\mathsf{Recon}(\mathbf{Q}, \mathbf{A}, \mathbf{H}) = \mathbf{D}[\mathbf{Q}]\right] \geq \varepsilon,$$

where $\mathbf{D} \xleftarrow{\$} \{0,1\}^n$, $\mathbf{Q} \xleftarrow{\$} [n]^k$ are uniform and independent random variables, and $\mathbf{A} \xleftarrow{\$} \mathsf{Advice}(\mathbf{D})$ and $\mathbf{H} \xleftarrow{\$} \mathsf{Hint}(\mathbf{D}, \mathbf{Q}, \mathbf{A})$ are the outputs of the respective algorithms on the random inputs.

First, we note that dual PIR schemes are very easy in the case that $s \geq k$. In this setting, the client can simply ignore the t-bit transcript from the Advice algorithm and simply privately

retrieve the k queried entries, $D[Q]$ in Hint. Therefore, we focus on the setting of $s < k$ where this construction is not possible.

One may wonder the formal relations between dual PIR and PIR with preprocessing. In a sense, dual PIR provides the client both more and less information and flexibility at different steps of the protocol compared to PIR with preprocessing. Nevertheless, we show that any PIR with preprocessing may be converted into a dual PIR scheme.

4.2 Constructing a Dual PIR

This section proves that any PIR with preprocessing scheme $\Pi^{\mathsf{O}} = (\mathsf{Init}, \mathsf{Setup}^{\mathsf{O}}, \mathsf{Query}^{\mathsf{O}})$ with an oracle-free encoding[6] (see Definition 3.8) that has access to a crypto oracle O that is sufficiently private and correct can be used to build a correct dual PIR $\Pi' = (\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}, \mathsf{Hint}^{\mathsf{O},\mathsf{RO}}, \mathsf{Recon}^{\mathsf{O},\mathsf{RO}})$ with access to both a crypto oracle O as well as an additional independent random oracle RO. We make the following assumptions about Π^{O} without loss of generality. First, we will consider underlying PIR schemes where the client storage is always exactly s bits (one can always pad to make sure the client storage is s bits). Both Init and $\mathsf{Setup}^{\mathsf{O}}$ will be randomized.[7] For $\mathsf{Query}^{\mathsf{O}}$, we will suppose that the server's execution during $\mathsf{Query}^{\mathsf{O}}$ is a deterministic function of its input (including the crypto oracle O, encoded database E and communication transcript τ). We note that assuming the server execution of $\mathsf{Query}^{\mathsf{O}}$ is deterministic is without loss of generality as Init is randomized and may attach the necessary server randomness to respond to the k queries into the encoded database E to be used during $\mathsf{Query}^{\mathsf{O}}$ by the server. However, the client may use randomness in its execution of $\mathsf{Query}^{\mathsf{O}}$.

In our dual PIR construction, we will consider executing the client-side of the Query algorithm of our underlying PIR scheme Π. As a reminder, the Query algorithm is a two-party, interactive protocol (potentially consisting of multiple rounds) between the client with input client state st and query sequence Q, and a server with the encoded database E (see Definition 3.7 for more details). We denote the execution of this algorithm by $(\mathsf{ans}; \perp \mid \tau) \leftarrow \mathsf{Query}^{\mathsf{O}}(\mathsf{st}, Q; E)$. The output consists of a private output ans to the client and a shared output τ consisting of the communication transcript between both parties ($\perp$ denotes that there is no private output to the server). We will also consider the transcript τ into two components consisting of communication sent by the client τ_{Client} and the communication sent by the server τ_{Server}. We denote the client-side execution of the query algorithm by the $(\mathsf{ans} \mid \tau_{\mathsf{Client}}) \leftarrow \mathsf{Query}^{\mathsf{O}}_{\mathsf{Client}}(\mathsf{st}, Q; \tau_{\mathsf{Server}})$ where we simulate the server-side computation by using its communication transcript denoted by τ_{Server}. The output is both the client's output ans along with the client's communication transcript τ_{Client}.

Overview of Dual PIR Construction. At a high level, $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}(D)$ will sample a communication transcript from the proper execution of underlying PIR construction. However, we note that $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}(D)$ only receives a database D and, in particular, does not receive the query Q. Instead, $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$ is parameterized by some fixed query sequence $Q' \in [n]^k$ and executes Π using the fixed query sequence Q' to obtain a communication transcript τ. Additionally, we will record all crypto oracle queries performed by the server in the set S. The final output of Advice will be $A = (\tau, S)$ consisting of the sampled communication transcript τ and all crypto oracle queries performed by the server S during the Query algorithm.

[6] We discuss plausible relaxations of this restriction in Remark 4.10.

[7] We note that randomized Init enables us to capture schemes with randomized (but oracle-free) encodings, as well as the derandomization of the server described next.

In the next subroutine of dual PIR, $\mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q, A = (\tau, S))$ receives the database D, the real query sequence of interest Q and the output of Advice denoted by $A = (\tau, S)$. As a note, Hint accepts any query sequences Q including those that are different from the fixed query sequence used in Advice such that $Q \neq Q'$. With this input, $\mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q, A = (\tau, S))$ will essentially re-randomize the client's execution of Π using input query sequence Q while fixing the server's execution. In more detail, we will iterate through all possible choices of randomness (including randomness in Setup and the choice of crypto oracle) to find some client state st such that the client's execution matches the communication transcript τ. To fix the server's execution, we sample crypto oracles $\mathsf{O}_1, \ldots, \mathsf{O}_z$ such that each sampled crypto oracle matches O for all queries in S for some choice of z we will fix later. For sampling each crypto oracle O_i that is consistent with $\mathsf{O}(S)$, we use the algorithm described in Lemma 3.6 with the shared random oracle RO. As we assume without loss of generality that the server's algorithm is deterministic, fixing all its server crypto oracle results means that the server's execution is fixed now. Note, Hint may obtain the same encoded database E by executing $\mathsf{Init}(D)$ using the same randomness $\omega_{\mathsf{Server}} = \mathsf{RO}(0)$ as used in Advice; as Init makes no oracle queries, this recomputed E is identical to the one in Advice and, importantly, does not depend on the crypto oracle at all. Next, Hint will attempt to sample a client state using the z different crypto oracles. For all $i \in \{1, \ldots, z\}$, Hint samples a client state $(\mathsf{st}_i; \bot) \leftarrow \mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E)$ and checks whether the execution of the query algorithm $(\mathsf{ans}; \bot \mid \tau_i') \leftarrow \mathsf{Query}^{\mathsf{O}_i}(\mathsf{st}, Q; E)$ produces the same communication transcript $\tau = \tau_i'$. For the first such instance with index i, Hint will encode the sampled client state st_i and crypto oracle O_i in its output H. We will revisit this encoding in more detail later as encoding O_i is clearly not efficient, but we will assume this is possible for now.

Finally, $\mathsf{Recon}^{\mathsf{O},\mathsf{RO}}(Q, A, H)$ executes the query algorithm of the underlying PIR protocol using the encoded communication transcript τ from Advice as well as the client state st and crypto oracle O_i from Hint. As a reminder, Recon does not receive the database D as input. Instead, it executes only the client-side of the query algorithm $(\mathsf{ans} \mid \tau_{\mathsf{Client}}) \leftarrow \mathsf{Query}^{\mathsf{O}_i}_{\mathsf{Client}}(\mathsf{st}, Q; \tau_{\mathsf{Server}})$ where the server's interaction is replicated using the server's communication in the transcript τ_{Server}. The output of Recon will be the answer ans obtained by executing the client-side of the query algorithm.

Encoding Algorithm for Hint**.** Now, we revisit the issue of encoding a crypto oracle O_i for Hint. While the client state st_i is s bits, the crypto oracle O_i is much larger and cannot be encoded efficiently. As a reminder, our high level goal is to show that there exists a dual PIR where the output of Hint is $O(s)$ bits. A first attempt might be for Hint to simply encode the sampled client state st. As the crypto oracles $\mathsf{O}_1, \ldots, \mathsf{O}_z$ are sampled using a shared random oracle RO, Recon can also sample them without any additional information. Afterwards, Recon could attempt to execute the query algorithm using st for each of the crypto oracles and find the first crypto oracle O_j that results in the communication transcript τ. Unfortunately, there is no guarantee that Recon ends up at the same oracle as Hint such that $j = i$. In particular, Recon does not receive the database. Therefore, Recon cannot determine whether st_i is a possible output of $\mathsf{Setup}^{\mathsf{O}_j}(1^\lambda; E)$. For example, it is a possible st_i somehow results in a communication matching τ for some index $j < i$ where st_i cannot be output by $\mathsf{Setup}^{\mathsf{O}_j}(1^\lambda; E)$. Another idea might be to instead encode the index i directly. Unfortunately, it turns out that the index i might need to be quite large since Hint might need to use $z = 2^{\Omega(|\tau|)}$ iterations to find a sampled client state and crypto oracle that results in the communication transcript τ output by Advice. This would be necessary if the distribution of the transcripts is uniformly random over all $|\tau|$-bit strings.

To resolve this, we use a combination of the above two ideas with additional changes to fix

various issues that arise in both approaches. To ensure that both Hint and Recon use the same client state in each execution, we will instead sample client state and randomness $(\mathsf{st}_i, \omega_{\mathsf{Client},i}) \leftarrow \mathsf{RO}_0(i)$ to be used with crypto oracle O_i using the shared random oracle RO. We also modify Hint such that it will also honestly generate a client state $(\mathsf{st}'_i; \perp) \leftarrow \mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E)$ where E is the encoded database from $E \leftarrow \mathsf{Init}(1^\lambda, D)$ using randomness $\omega_{\mathsf{Server}} = \mathsf{RO}(0)$. If $\mathsf{st}_i \neq \mathsf{st}'_i$, then Hint will essentially skip past using the i-th client state and crypto oracle. However, Hint will still execute the query algorithm using the random client state st_i, client randomness $\omega_{\mathsf{Client},i}$, and crypto oracle O_i to obtain the communication transcript $(\mathsf{ans}; \perp \mid \tau_i) \leftarrow \mathsf{Query}^{\mathsf{O}_i}(\mathsf{st}_i, Q; E)$ using $\omega_{\mathsf{Client},i}$ as the client-side randomness. Hint will keep a counter c of the number of times that $\tau_i = \tau$ matching the transcript output by Advice. With this change, Hint will find the first index $i \in [z]$ such that the randomly sampled client state and randomness $(\mathsf{st}_i, \omega_{\mathsf{Client},i})$ using the random oracle RO matches the honestly generated client state $\mathsf{st}'_i \leftarrow \mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E)$ and the resulting communication transcript matches τ. Instead of encoding the index i, Hint will instead encode the counter c keeping track of the number of times it has observed transcript matches (that is, Query outputs the same communication transcript as τ received from Advice) before finding an iteration where the observed transcript matches and Setup gives the same client state as the one sampled by using RO. We will later show that this counter c only requires $O(s)$ bits except with negligible probability. It turns out that this counter is sufficient for Recon to reconstruct both the sampled client state st_i and crypto oracle O_i without needing the database D. In particular, Recon executes the client-side of the query algorithm for each of the sampled client states and crypto oracles and observes whether the output communication transcript matches τ output by Advice. On the c-th matching transcript occurrence, Recon will successfully decode the same client state st_i and crypto oracle O_i that Hint needed to encode.

Our Dual PIR Construction. We formally present the Dual PIR below. In the description below, Q' is a fixed set of k queries such as $Q' = (1, \ldots, k)$ whereas Q is an input. For convenience, we will abuse notation and consider the outputs of the random oracle RO to be variable length depending on the desired output. For random oracles RO with fixed output lengths, one can always add unique prefixes/suffixes to derive arbitrary amounts of randomness. Similarly, we assume the inputs to RO are domain separated: the input 0 used to derive $\omega_{\mathsf{Server}} = \mathsf{RO}(0)$ is encoded distinctly from every input of the form $i \mid\mid x$ used by the virtual oracles $\mathsf{RO}_0, \ldots, \mathsf{RO}_z$ (e.g., via a dedicated prefix), so the randomness of ω_{Server} is independent of all other randomness derived from RO.

1. $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$:
   - Run $E \leftarrow \mathsf{Init}(1^\lambda, D)$ using $\omega_{\mathsf{Server}} = \mathsf{RO}(0)$ as the algorithm's randomness.
   - Run $(\mathsf{st}; \perp) \leftarrow \mathsf{Setup}^{\mathsf{O}}(1^\lambda; E)$.
   - Run $(\mathsf{ans}; \perp \mid \tau) \leftarrow \mathsf{Query}^{\mathsf{O}}(\mathsf{st}, Q'; E)$.
   - Record all server-side oracle O queries executed during the Query algorithm as S.
   - Output $A \leftarrow (\tau, S)$.
2. $\mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q, A)$:
   - Parse $A = (\tau, S)$.
   - Compute $\mathsf{O}(S) \leftarrow \{(q, \mathsf{O}(q)) \mid q \in S\}$.

- Run $E \leftarrow \mathsf{Init}(1^\lambda, D)$ using $\omega_{\mathsf{Server}} = \mathsf{RO}(0)$ as the algorithm's randomness.
- Set $z \leftarrow 2^{3n+2s}$.
- Construct virtual random oracles $\mathsf{RO}_0, \mathsf{RO}_1, \ldots, \mathsf{RO}_z$ such that $\mathsf{RO}_i(\cdot) = \mathsf{RO}(i \mid\mid \cdot)$ and $i \in \{0, 1, \ldots, z\}$ is represented using a string of $O(n+s)$ bit length.
- Instantiate counter $c \leftarrow 0$.
- For $i \in \{1, \ldots, z\}$ and while $c < 2^{3s}$:
  (a) Sample oracle $\mathsf{O}_i \leftarrow \mathcal{A}^{\mathsf{RO}_i}(\mathsf{O}(S))$ such that $\mathsf{O}_i(S) = \mathsf{O}(S)$ using Lemma 3.6 with parameter $\zeta = 6n$. If the algorithm fails to sample crypto oracle O_i, then output $H \leftarrow \perp$.
  (b) Compute $(\mathsf{st}_i, \omega_{\mathsf{Client},i}) \leftarrow \mathsf{RO}_0(i)$.
  (c) Compute $(\mathsf{st}'_i; \perp) \leftarrow \mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E)$.
  (d) Compute $(\mathsf{ans}; \perp \mid \tau_i) \leftarrow \mathsf{Query}^{\mathsf{O}_i}(\mathsf{st}_i, Q; E)$ using $\omega_{\mathsf{Client},i}$ as the randomness for the client side of the Query protocol.
  (e) If $\tau_i = \tau$:
    i. If $\mathsf{st}_i = \mathsf{st}'_i$, output $H \leftarrow c$.
    ii. Increment $c \leftarrow c + 1$.
- Output $H \leftarrow \perp$.

3. $\mathsf{Recon}^{\mathsf{O},\mathsf{RO}}(Q, A, H)$:

- Parse $A = (\tau, S)$.
- Parse $H = c$.
- Parse $\tau = (\tau_{\mathsf{Client}}, \tau_{\mathsf{Server}})$ as client and server communication respectively.
- Compute $\mathsf{O}(S) \leftarrow \{(q, \mathsf{O}(q)) \mid q \in S\}$.
- Set $z \leftarrow 2^{3n+2s}$.
- Construct virtual random oracles $\mathsf{RO}_0, \mathsf{RO}_1, \ldots, \mathsf{RO}_z$ such that $\mathsf{RO}_i(\cdot) = \mathsf{RO}(i \mid\mid \cdot)$ and $i \in \{0, 1, \ldots, z\}$ is represented using string of $O(n+s)$ bits.
- Instantiate counter $c' \leftarrow 0$.
- For $i \in \{1, \ldots, z\}$:
  (a) Sample oracle $\mathsf{O}_i \leftarrow \mathcal{A}^{\mathsf{RO}_i}(\mathsf{O}(S))$ such that $\mathsf{O}_i(S) = \mathsf{O}(S)$ using Lemma 3.6 with parameter $\zeta = 6n$. If the algorithm fails to sample crypto oracle O_i, then output $R \leftarrow \perp$.
  (b) Compute $(\mathsf{st}, \omega_{\mathsf{Client}}) \leftarrow \mathsf{RO}_0(i)$.
  (c) Compute $(\mathsf{ans} \mid \tau'_{\mathsf{Client}}) \leftarrow \mathsf{Query}^{\mathsf{O}_i}_{\mathsf{Client}}(\mathsf{st}, Q; \tau_{\mathsf{Server}})$ using ω_{Client} as the randomness for the client side of the Query protocol.
  (d) If $\tau'_{\mathsf{Client}} = \tau_{\mathsf{Client}}$:
    i. If $c' = c$, output $R \leftarrow \mathsf{ans}$.
    ii. Increment $c' \leftarrow c' + 1$.
- Output $R \leftarrow \perp$.

4.3 Correctness Analysis of Dual PIR Construction

This section will be devoted to proving the correctness of our dual PIR construction. In particular, we will prove the following theorem:

Theorem 4.3. *Suppose the PIR construction* $\Pi = (\mathsf{Init}, \mathsf{Setup}^{\mathsf{O}}, \mathsf{Query}^{\mathsf{O}})$ *has an oracle-free encoding (Definition 3.8), supports batch queries of size* k *for databases of size* n *with client state of* s *bits, the communication transcript and all server crypto oracle queries during the query algorithm are* t *bits in total. Furthermore, suppose* Π *is* ε*-correct and* δ*-private. For any fixed query sequence* Q'*, our construction is a* $(n, k, t, 3s)$*-dual PIR with* ε'*-correctness satisfying* $\varepsilon' \geq \varepsilon - 2\delta - 2^{-s+1} - 2^{-n+1}$.

To prove the correctness of our dual PIR construction, we will use the following steps:

1. First, we analyze our dual PIR construction when we assume that the input query sequence Q to both Hint and Recon is the same as the fixed query sequence Q' used in Advice. To do this, we prove the following properties about our dual PIR construction when $Q = Q'$:
   (a) Hint successfully samples a client state st_i and crypto oracle O_i resulting in a communication transcript matching the output of Advice (and, thus Hint outputs a counter $c \neq \perp$) except with probability exponentially small in s.
   (b) If Hint outputs a valid counter $c \neq \perp$, then the resulting output distribution of Recon is equivalent to a proper execution of the underlying PIR on input database D, query sequence Q' and a properly sampled crypto oracle O. As a result, Recon successfully obtains the correct answer $D[Q']$ with probability similar to the correctness probability of the underlying PIR.
2. As the last step, we prove that our dual PIR must be correct for any query sequence $Q \neq Q'$. If this is not the case, we show there exists an adversary that breaks the underlying PIR privacy using the same number of oracle queries as an honest server.

Analyzing Output of Hint**.** To start, we will prove several properties about the output of Hint. First, we denoted the output of $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$ by $A = (\tau, S)$. Throughout this section, we fix the encoding randomness $\omega_{\mathsf{Server}} = \mathsf{RO}(0)$ and, thus, the encoded database $E = \mathsf{Init}(1^\lambda, D)$ computed using randomness ω_{Server}. As Init makes no oracle queries, E is a fixed value that is identical in Advice, Hint, and the honest PIR execution, and is independent of the crypto oracle O. All probability statements below hold for any fixed choice of ω_{Server} (equivalently, of E).

Consider the following *resampled experiment* associated with the iterations of Hint: first sample a crypto oracle O_i that is consistent with $\mathsf{O}(S)$, then a client state $\mathsf{st}' \leftarrow \mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E)$, and execute $\mathsf{Query}^{\mathsf{O}_i}(\mathsf{st}', Q'; E)$ with fresh client randomness to obtain a communication transcript τ_i; finally, check whether $\tau_i = \tau$. (The iterations of Hint instead run Query with a *uniformly* sampled state; as we show in Lemma 4.6, the experiment above with an honestly sampled state characterizes both the distribution of the advice and the acceptance rate of Hint.) For a set of query-answer pairs $v = \{(q, v(q)) \mid q \in S\}$, we denote the probability of this event by

$$p(\tau, S, v, E) = \Pr[\boldsymbol{\tau} = \tau]$$

over the random choice of O_i conditioned on $\mathsf{O}_i(S) = v$ and the internal randomness of Setup and Query (noting that the client-side randomness ω_{Client} for Query is derived explicitly from RO_0 in Hint, but p averages over all such choices), where $\boldsymbol{\tau} \leftarrow \mathsf{Query}^{\mathsf{O}_i}(\mathbf{st}, Q'; E)$ and $\mathbf{st} \leftarrow \mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E)$.

Before analyzing Hint, we record two consequences of the fact that the fixed encoded database E is independent of the crypto oracle. The first states that, once an oracle agrees with O on S, the entire server-side execution of Query is replayed exactly. The second states that the transcript distribution of an honest execution, conditioned on the server's oracle queries and answers, is exactly the transcript distribution of the resampled executions performed by Hint. The second part is precisely where the oracle-free encoding assumption is used: if Init could query O, then E would be correlated with O outside of S, and an honestly generated pair (E, O) would not be distributed as the pair (E, O_i) used in the resampled executions of Hint.

Claim 4.4 (Server Replay and Advice Consistency)**.** *Fix any database D, query sequence Q', encoding randomness ω_{Server} (and thus E), transcript $\tau = (\tau_{\mathsf{Client}}, \tau_{\mathsf{Server}})$, query set S, and answers v such that (τ, S) is a possible output of $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$ when $\mathsf{O}(S) = v$. Then:*

1. (Server replay.) *For every crypto oracle O^* such that $\mathsf{O}^*(S) = v$, every client state st, every client randomness ω_{Client}, and every query sequence Q: in the execution $\mathsf{Query}^{\mathsf{O}^*}(\mathsf{st}, Q; E)$, for every round r, if the client's first r messages equal the first r messages of τ_{Client}, then the server's oracle queries, answers, and messages through round r equal the corresponding prefix of the execution recorded by Advice (in particular, its queries lie in S with answers given by v, and its messages equal the corresponding prefix of τ_{Server}). Consequently, if the client sends all of τ_{Client}, then the server makes exactly the oracle queries S (in the same order), receives the answers v, and sends the messages τ_{Server}; the execution produces the full transcript τ if and only if the client-side messages equal τ_{Client}; and any execution under O^* with transcript τ has server query set exactly S.*

2. (Advice consistency.) *For the honest execution inside $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$ over a uniformly sampled crypto oracle O,*

$$\Pr\left[(\boldsymbol{\tau}, \mathbf{S}) = (\tau, S) \wedge \mathsf{O}(S) = v\right] = \Pr\left[\mathsf{O}(S) = v\right] \cdot p(\tau, S, v, E).$$

Proof. For (1), recall that the server's execution during Query is a deterministic function of the encoded database E, the incoming client messages, and its oracle answers. The server's execution recorded by Advice (with an oracle answering v on S, encoded database E, and incoming client messages τ_{Client}) made the queries S in some order, received the answers v, and produced the messages τ_{Server}. Consider any execution under O^* with the same encoded database E (recall that E does not depend on the choice of oracle since Init makes no oracle queries) in which the client sends the messages τ_{Client}. The server's first oracle query is determined by E and a prefix of τ_{Client}, so it is the same first query $q_1 \in S$, and the answer $\mathsf{O}^*(q_1) = v(q_1)$ also matches. By induction, every subsequent query, answer, and outgoing message of the server matches the execution recorded by Advice, proving the first statement. The remaining statements follow immediately: if the client sends τ_{Client}, the server sends τ_{Server}, so the transcript is τ if and only if the client-side messages equal τ_{Client}, in which case the query set is exactly S.

For (2), note that conditioned on the event $\mathsf{O}(S) = v$, the uniformly sampled crypto oracle O is distributed exactly as the resampled oracles O_i (both are a uniformly random crypto oracle conditioned on answering v on S; see Lemma 3.6). Moreover, the honest execution inside Advice touches the oracle only through Setup and Query using the fixed E (again, since Init makes no oracle queries), which is exactly the experiment defining $p(\tau, S, v, E)$, except that the event of p does not require the server query set to equal S. By part (1), any execution under an oracle agreeing with v on S that produces transcript τ has server query set exactly S, so the two events coincide and $\Pr[(\boldsymbol{\tau}, \mathbf{S}) = (\tau, S) \mid \mathsf{O}(S) = v] = p(\tau, S, v, E)$. □

We now show that Advice will output a pair $A = (\tau, S)$ such that this probability p is at least 2^{-3n} except with probability negligible in n.

Lemma 4.5. *Suppose the underlying PIR protocol has $o(n)$ bits of communication and the server performs blackbox crypto operations on at most $o(n)$ bits. For any choice of database D and query sequence Q', let $\mathbf{p} = p(\boldsymbol{\tau}, \mathbf{S}, \mathsf{O}(\mathbf{S}), E)$ be the above probability where $(\boldsymbol{\tau}, \mathbf{S}) \leftarrow \mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$. Then,*

$$\Pr[\mathbf{p} \geq 2^{-3n}] \geq 1 - 2^{-n}.$$

over the internal randomness of Advice *and random choice of* O.

Proof. Consider the set Z consisting of all triples (τ, S, v) such that $p(\tau, S, v, E) < 2^{-3n}$. Using part (2) of Claim 4.4,

$$\begin{aligned}\Pr[\mathbf{p} < 2^{-3n}] &= \sum_{(\tau,S,v)\in Z} \Pr[(\boldsymbol{\tau}, \mathbf{S}) = (\tau, S) \wedge \mathsf{O}(S) = v] \leq \sum_{(\tau,S,v)\in Z} \Pr[\mathsf{O}(S) = v] \cdot p(\tau, S, v, E) \\ &< 2^{-3n} \cdot \sum_{(\tau,S)} \sum_{v} \Pr[\mathsf{O}(S) = v] = 2^{-3n} \cdot \sum_{(\tau,S)} 1 \leq 2^{-3n} \cdot 2^{o(n)} \leq 2^{-n},\end{aligned}$$

where the first sum effectively ranges over the triples in Z that arise with positive probability as advice outcomes (to which part (2) of Claim 4.4 applies; all other summands on the left vanish, and enlarging the second sum to all of Z only increases the bound). We also use that $\sum_v \Pr[\mathsf{O}(S) = v] = 1$ for every fixed query set S, and that the total number of pairs (τ, S) is at most $2^{|\tau|+|S|} = 2^{o(n)}$ since the underlying PIR protocol has $|\tau| = o(n)$ communication and the server performs blackbox cryptography operations on $|S| = o(n)$ bits. □

In other words, we have shown that by executing $1/p(\boldsymbol{\tau}, \mathbf{S}, \mathsf{O}(\mathbf{S}), E) \leq 2^{3n}$ sampling experiments in Hint will successfully sample a crypto oracle O_i and client state st'_i that will result in this communication transcript. Next, we show that Hint outputs $H = c$ such that $c \neq \perp$ except with negligible probability. To do this, we essentially show the number of erroneous times that the communication transcript matches the one output by Advice is essentially equivalent to $1/p(\boldsymbol{\tau}, \mathbf{S}, \mathsf{O}(\mathbf{S}), E)$.

Lemma 4.6. *Suppose the underlying PIR protocol has $o(n)$ bits of communication and the server performs blackbox crypto operations on at most $o(n)$ bits. For any choice of database D and query sequence Q',*

$$\Pr[\mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q', (\boldsymbol{\tau}, \mathbf{S})) = \perp \mid (\boldsymbol{\tau}, \mathbf{S}) \leftarrow \mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)] \leq 2^{-s+1} + 2^{-n+1}$$

over the random choices of O *and* RO *as well as the internal randomness of* Advice.

Proof. We start by analyzing the probability that Hint outputs $\perp$ due to the failure of sampling a crypto oracle. As a reminder, Hint samples 2^{3n+2s} crypto oracles using the algorithm in Lemma 3.6 with parameter $\zeta = 6n = \mathrm{poly}(\lambda)$ since $n = \mathrm{poly}(\lambda)$. Therefore, a single crypto oracle sample fails with probability e^{-6n}. By a union bound, the probability that any crypto oracle sample fails is at most $2^{3n+2s} \cdot e^{-6n} \leq 2^{5n-6n} = 2^{-n}$. For the remainder of the proof, we condition on all crypto oracle samples succeeding; by Lemma 3.6, each O_i is then an independent, uniformly random crypto oracle conditioned on $\mathsf{O}_i(S) = \mathsf{O}(S)$.

Next, we identify the two remaining ways in which Hint can output $\perp$. We will consider the first $z' = \lfloor 2^{2s}/p(\boldsymbol{\tau}, \mathbf{S}, \mathsf{O}(\mathbf{S}), E) \rfloor \leq 2^{2s} \cdot 2^{3n} = z$ iterations of the loop, where $(\boldsymbol{\tau}, \mathbf{S})$ is the output

of $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$ and we assume $p(\boldsymbol{\tau}, \mathbf{S}, \mathsf{O}(\mathbf{S}), E) \geq 2^{-3n}$, adding the failure probability 2^{-n} from Lemma 4.5 at the end. Say that iteration i is a *match* if $\tau_i = \tau$, and an *acceptance* if additionally $\mathsf{st}_i = \mathsf{st}'_i$. If some iteration $i \leq z'$ is an acceptance and the number of matches strictly before it is fewer than 2^{3s}, then Hint outputs a counter $c \neq \perp$. Therefore, denoting by $\mathbf{c}$ the number of matches in the first z' iterations,

$$\Pr[\mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q', (\boldsymbol{\tau}, \mathbf{S})) = \perp] \leq \Pr[\mathbf{c} \geq 2^{3s}] + \Pr[\text{no acceptance in } z' \text{ iterations}] + 2^{-n} + 2^{-n}.$$

We first bound $\mathbb{E}[\mathbf{c}]$. Fix an advice output and oracle answers (τ, S, v) where $v = \mathsf{O}(S)$. Each iteration i independently samples O_i (uniform conditioned on $\mathsf{O}_i(S) = v$) and a uniform pair $(\mathsf{st}_i, \omega_{\mathsf{Client},i}) \leftarrow \mathsf{RO}_0(i)$, all independent of the advice execution (recall that RO is independent of O and, by domain separation, the values $\mathsf{RO}_0(i)$ and the randomness of each virtual oracle RO_i are independent of ω_{Server}). Therefore, every iteration is a match independently with the same probability

$$q(\tau, S, v) := \Pr\left[\mathsf{Query}^{\mathsf{O}_i}(\mathsf{st}_i, Q'; E) \text{ produces transcript } \tau\right]$$

over the choice of O_i conditioned on $\mathsf{O}_i(S) = v$ and uniform $(\mathsf{st}_i, \omega_{\mathsf{Client},i})$. Note that q does not depend on i and differs from p only in that the client state is uniformly random instead of honestly sampled using $\mathsf{Setup}^{\mathsf{O}_i}$. Using part (2) of Claim 4.4 and linearity of expectation,

$$\begin{aligned}\mathbb{E}[\mathbf{c}] &\leq \sum_{(\tau,S,v)} \Pr[(\boldsymbol{\tau}, \mathbf{S}) = (\tau, S) \wedge \mathsf{O}(S) = v] \cdot \frac{2^{2s}}{p(\tau, S, v, E)} \cdot q(\tau, S, v) \\ &= 2^{2s} \sum_{(\tau,S,v)} \Pr[\mathsf{O}(S) = v] \cdot q(\tau, S, v) \leq 2^{2s},\end{aligned}$$

where both sums effectively range over the triples (τ, S, v) arising with positive probability as advice outcomes (all other summands in the first line vanish), to which Claim 4.4 applies, and the final inequality holds as follows. Note, sampling a uniformly random crypto oracle conditioned on answering v on S and multiplying by $\Pr[\mathsf{O}(S) = v]$ is equivalent to sampling an unconditioned uniformly random crypto oracle and restricting to the event $\mathsf{O}_i(S) = v$. Combined with part (1) of Claim 4.4 (any execution under an oracle agreeing with v on S with transcript τ has server query set exactly S), we get that

$$\Pr[\mathsf{O}(S) = v] \cdot q(\tau, S, v) = \Pr\left[\mathsf{Query}^{\mathsf{O}_i}(\mathsf{st}_i, Q'; E) \text{ produces } \tau \text{ with query set } S \text{ and } \mathsf{O}_i(S) = v\right]$$

where the probability is now over an *unconditioned* uniformly random crypto oracle O_i and uniform $(\mathsf{st}_i, \omega_{\mathsf{Client},i})$. A single execution determines a unique triple (its transcript, its server query set, and the oracle's answers on that query set), so these events are disjoint across distinct triples (τ, S, v) and their probabilities sum to at most 1. By Markov's inequality, $\Pr[\mathbf{c} \geq 2^{3s}] \leq 2^{-s}$.

Finally, we bound the probability that no acceptance occurs in the first z' iterations. In each iteration, the state st_i is uniform over $\{0,1\}^s$ (recall client states are padded to exactly s bits) and is independent of the honestly sampled state $\mathsf{st}'_i \leftarrow \mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E)$. Therefore, each iteration is an acceptance independently with probability exactly

$$\mathbb{E}_{\mathsf{O}_i}\left[\sum_{\mathsf{st} \in \{0,1\}^s} 2^{-s} \cdot \Pr\left[\mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E) = \mathsf{st}\right] \cdot \Pr\left[\mathsf{Query}^{\mathsf{O}_i}(\mathsf{st}, Q'; E) \text{ produces } \tau\right]\right] = 2^{-s} \cdot p(\tau, S, v, E),$$

since requiring the uniform state st_i to equal an independent honest sample st'_i is equivalent to using the honest sample directly, at an exact multiplicative 2^{-s} loss in probability. Over the $z' = \lfloor 2^{2s}/p \rfloor$ independent iterations, the probability that no acceptance occurs is at most

$$\left(1 - 2^{-s} \cdot p\right)^{z'} \leq \left(1 - 2^{-s} \cdot p\right)^{2^{2s}/p - 1} \leq 2 \cdot e^{-2^s} \leq 2^{-2^s+1} \leq 2^{-s},$$

using $(1 - 2^{-s} \cdot p)^{-1} \leq 2$ and $2^s \geq s + 1$ for $s \geq 1$. Altogether, Hint outputs $\perp$ with probability at most $2^{-s} + 2^{-s} + 2^{-n} + 2^{-n} = 2^{-s+1} + 2^{-n+1}$. □

Correctness of Recon**.** Finally, we show that Recon outputs the correct answer of $D[Q']$ with high probability assuming that Hint outputs a counter $c \neq \perp$. To do this, we essentially show that whenever $c \neq \perp$, the distribution of the output of Recon is identical to a proper execution of the underlying PIR. As a result, the correctness probability of Recon will be the same as the ε-correctness of the underlying PIR.

Lemma 4.7. *Fix any database D, query sequence Q', and encoding randomness ω_{Server} (thus, the encoded database E). For every transcript, query set, and answer assignment (τ, S, v) occurring with positive probability in the honest execution below, the following two distributions are identical:*

1. *The distribution $\mathcal{D}$ of $(\mathbf{st}, \mathbf{ans})$ resulting from the execution of the dual PIR where $(\boldsymbol{\tau}, \mathbf{S}) \leftarrow \mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$, $\mathbf{c} \leftarrow \mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q', (\boldsymbol{\tau}, \mathbf{S}))$ and $\mathbf{ans} \leftarrow \mathsf{Recon}^{\mathsf{O},\mathsf{RO}}(Q', (\boldsymbol{\tau}, \mathbf{S}), \mathbf{c})$, conditioned on the events $(\boldsymbol{\tau}, \mathbf{S}) = (\tau, S)$, $\mathsf{O}(S) = v$, and $\mathbf{c} \neq \perp$, where $\mathbf{st}$ denotes the client state st_i of the accepting iteration of* Hint*.*
2. *The distribution $\mathcal{D}'$ of $(\mathbf{st}', \mathbf{ans}')$ resulting from the proper execution of the underlying PIR where $\mathbf{st}' \leftarrow \mathsf{Setup}^{\mathsf{O}}(1^\lambda; E)$, $(\mathbf{ans}' \mid \boldsymbol{\tau}') \leftarrow \mathsf{Query}^{\mathsf{O}}(\mathbf{st}', Q'; E)$ and $\mathbf{S}'$ is the set of all server oracle O queries performed during* Query*, conditioned on the event $\{(\boldsymbol{\tau}', \mathbf{S}') = (\tau, S) \wedge \mathsf{O}(S) = v\}$.*

Moreover, conditioned on the event in the first distribution, Recon *outputs exactly the answer* **ans** *of the client in the accepting iteration of* Hint*. This implies that*

$$\begin{aligned}\Pr[\mathbf{ans} = D[Q'] \mid (\boldsymbol{\tau}, \mathbf{S}) = (\tau, S), \mathsf{O}(S) = v, \mathbf{c} \neq \perp] \\ = \Pr[\mathbf{ans}' = D[Q'] \mid (\boldsymbol{\tau}', \mathbf{S}') = (\tau, S), \mathsf{O}(S) = v].\end{aligned}$$

Proof. We first show that Recon recovers the state, randomness, and oracle of the accepting iteration of Hint, and then show that the accepting iteration's client execution is distributed as an honest client execution conditioned on (τ, S, v).

We first claim that for every iteration index i, iteration i of Hint is a match (i.e., $\tau_i = \tau$) if and only if iteration i of Recon is a client-side match (i.e., $\tau'_{\mathsf{Client}} = \tau_{\mathsf{Client}}$). First, note that Hint and Recon construct identical iterations: the resampling algorithm of Lemma 3.6 is deterministic given RO_i and $\mathsf{O}(S)$, and both parties compute $\mathsf{O}(S)$ by querying the real oracle on the set S contained in the advice, so iteration i of both algorithms uses the same oracle O_i, state st_i, and client randomness $\omega_{\mathsf{Client},i}$ (and identical resampling failures). For the forward direction, if Hint's iteration i produces the full transcript τ, then the client's view in that execution consists of $(\mathsf{st}_i, \omega_{\mathsf{Client},i}, \mathsf{O}_i)$ and incoming server messages τ_{Server}; the client-side execution $\mathsf{Query}^{\mathsf{O}_i}_{\mathsf{Client}}(\mathsf{st}_i, Q'; \tau_{\mathsf{Server}})$ run by Recon has the identical view and therefore produces the identical client messages τ_{Client} and answer. For the reverse direction, suppose Recon's iteration i produces client messages τ_{Client} when fed the server

messages τ_{Server}. Consider Hint's interactive execution $\mathsf{Query}^{\mathsf{O}_i}(\mathsf{st}_i, Q'; E)$. We argue by induction over rounds that it reproduces the transcript τ: if the first $r-1$ rounds match τ, then the client's message in round r equals that of Recon's replay (same state, randomness, oracle, and incoming messages), which is the corresponding message of τ_{Client}; and the server's message in round r equals the corresponding message of τ_{Server} by part (1) of Claim 4.4, using $\mathsf{O}_i(S) = \mathsf{O}(S) = v$ and the identical encoded database E (again, since Init makes no oracle queries). Hence Hint's iteration i is a full match. It follows that the sequences of matching iterations in Hint and Recon coincide, so on input the counter c, Recon stops at exactly the accepting iteration of Hint and outputs the answer of that iteration's client execution.

Next, we show that the accepting iteration is distributed honestly. Condition on $(\boldsymbol{\tau}, \mathbf{S}) = (\tau, S)$ and $\mathsf{O}(S) = v$ (and, as in Lemma 4.6, on all resamplings succeeding). The iterations of Hint are independent and identically distributed: iteration i samples O_i uniformly conditioned on $\mathsf{O}_i(S) = v$, uniform $(\mathsf{st}_i, \omega_{\mathsf{Client},i})$, and fresh internal coins for $\mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E)$, all independent of the conditioning event. Iteration i is an acceptance if and only if the transcript matches τ and $\mathsf{st}_i = \mathsf{st}'_i$. For any oracle O^*, state st^*, and randomness ω^*, the probability that iteration i is an acceptance with $(\mathsf{O}_i, \mathsf{st}_i, \omega_{\mathsf{Client},i}) = (\mathsf{O}^*, \mathsf{st}^*, \omega^*)$ is

$$\Pr[\mathsf{O}_i = \mathsf{O}^*] \cdot 2^{-s} \cdot \Pr[\boldsymbol{\omega}_{\mathsf{Client},i} = \omega^*] \cdot \Pr\left[\mathsf{Setup}^{\mathsf{O}^*}(1^\lambda; E) = \mathsf{st}^*\right] \cdot \mathbb{1}\left[\mathsf{Query}^{\mathsf{O}^*}(\mathsf{st}^*, Q'; E) \text{ with } \omega^* \text{ produces } \tau\right],$$

where the factor $\Pr[\mathsf{Setup}^{\mathsf{O}^*}(1^\lambda; E) = \mathsf{st}^*]$ arises from the requirement $\mathsf{st}_i = \mathsf{st}'_i$ over Setup's fresh coins. On the other hand, in the honest execution, the oracle O conditioned on $\mathsf{O}(S) = v$ is distributed identically to O_i, and by part (1) of Claim 4.4 the event $\{(\boldsymbol{\tau}', \mathbf{S}') = (\tau, S)\}$ is exactly the event that the transcript is τ. Hence the conditional distribution of $(\mathsf{O}, \mathbf{st}', \boldsymbol{\omega}'_{\mathsf{Client}})$ given $\{(\boldsymbol{\tau}', \mathbf{S}') = (\tau, S) \wedge \mathsf{O}(S) = v\}$ assigns each $(\mathsf{O}^*, \mathsf{st}^*, \omega^*)$ probability proportional to

$$\Pr[\mathsf{O}_i = \mathsf{O}^*] \cdot \Pr[\boldsymbol{\omega}_{\mathsf{Client},i} = \omega^*] \cdot \Pr\left[\mathsf{Setup}^{\mathsf{O}^*}(1^\lambda; E) = \mathsf{st}^*\right] \cdot \mathbb{1}\left[\mathsf{Query}^{\mathsf{O}^*}(\mathsf{st}^*, Q'; E) \text{ with } \omega^* \text{ produces } \tau\right].$$

The two expressions differ only by the constant factor 2^{-s} (here we use that client states are padded to exactly s bits), so the accepted triple $(\mathsf{O}_i, \mathsf{st}_i, \omega_{\mathsf{Client},i})$, conditioned on acceptance, is distributed identically to $(\mathsf{O}, \mathbf{st}', \boldsymbol{\omega}'_{\mathsf{Client}})$ conditioned on the above event. Finally, since the iterations are independent and identically distributed, the triple at the *first* acceptance has this same distribution, and further conditioning on the events defining $\mathbf{c} \neq \perp$ (that the first acceptance occurs within the loop bounds with fewer than 2^{3s} preceding matches) does not change this distribution, as those events depend only on the other (independent) iterations and on whether the accepting iteration accepts.

Combining the above, conditioned on $\{(\boldsymbol{\tau}, \mathbf{S}) = (\tau, S), \mathsf{O}(S) = v, \mathbf{c} \neq \perp\}$, the accepting iteration's triple $(\mathsf{O}_i, \mathsf{st}_i, \omega_{\mathsf{Client},i})$ is distributed as $(\mathsf{O}, \mathbf{st}', \boldsymbol{\omega}'_{\mathsf{Client}})$ in the honest execution conditioned on $\{(\boldsymbol{\tau}', \mathbf{S}') = (\tau, S), \mathsf{O}(S) = v\}$. In both executions, the client's answer is the same deterministic function of the triple (together with the fixed E and Q'): the full interactive execution of Query is determined by the triple and E (the server being deterministic). As shown above, Recon outputs exactly the accepting iteration's answer. Therefore, $(\mathbf{st}, \mathbf{ans}) \sim \mathcal{D}$ and $(\mathbf{st}', \mathbf{ans}') \sim \mathcal{D}'$ are identically distributed, which implies the final claim. □

Using this fact, we can deduce that our dual PIR has the same correctness probability as the underlying PIR except with probability corresponding to the event that the counter $\mathbf{c}$ output by Hint is $\perp$.

Lemma 4.8. *Fix any database D and query sequence Q' and suppose that the underlying PIR is ε-correct. Consider a proper execution of the dual PIR to obtain $(\boldsymbol{\tau}, \mathbf{S}) \leftarrow \mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$, $\mathbf{c} \leftarrow \mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q', (\boldsymbol{\tau}, \mathbf{S}))$ and $\mathbf{ans} \leftarrow \mathsf{Recon}^{\mathsf{O},\mathsf{RO}}(Q', (\boldsymbol{\tau}, \mathbf{S}), \mathbf{c})$. Then,*

$$\Pr[\mathbf{ans} = D[Q']] \geq \varepsilon - 2^{-s+1} - 2^{-n+1}$$

over the random choice of oracles O and RO as well as the internal randomness of the algorithms.

Proof. Since Advice performs an honest execution of the underlying PIR using the fixed encoded database E, the joint distribution of $(\boldsymbol{\tau}, \mathbf{S}, \mathsf{O}(\mathbf{S}))$ in the dual PIR execution is identical to the joint distribution of $(\boldsymbol{\tau}', \mathbf{S}', \mathsf{O}(\mathbf{S}'))$ in a proper execution of the underlying PIR. Then, for every outcome (τ, S, v),

$$\begin{aligned}\Pr[\mathbf{ans} = D[Q'] \wedge \mathbf{c} \neq \perp \mid (\boldsymbol{\tau}, \mathbf{S}, \mathsf{O}(\mathbf{S})) = (\tau, S, v)] \\ \geq \Pr[\mathbf{ans}' = D[Q'] \mid (\boldsymbol{\tau}', \mathbf{S}', \mathsf{O}(\mathbf{S}')) = (\tau, S, v)] \\ - \Pr[\mathbf{c} = \perp \mid (\boldsymbol{\tau}, \mathbf{S}, \mathsf{O}(\mathbf{S})) = (\tau, S, v)],\end{aligned}$$

where we use Lemma 4.7 for the equality of the two conditional probabilities of outputting $D[Q']$ and that probabilities are at most 1. Averaging over the outcomes (τ, S, v) and the encoding randomness ω_{Server} (recall that ε-correctness averages over the coins of Init as well), and applying the ε-correctness of the underlying PIR together with Lemma 4.6, we conclude

$$\Pr[\mathbf{ans} = D[Q']] \geq \Pr[\mathbf{ans} = D[Q'] \wedge \mathbf{c} \neq \perp] \geq \varepsilon - 2^{-s+1} - 2^{-n+1}.$$

□

Dual PIR Correctness for All Queries. As the last step of the proof, we now show that our dual PIR construction is correct even when the input query sequence $Q \neq Q'$ that is given to both Hint and Recon does not match the fixed query sequence Q' used in $\mathsf{Advice}_{Q'}$. At a high level, we essentially show that, if the dual PIR is incorrect when given input query sequences $Q \neq Q'$, then we can construct an adversary that can break the privacy of the underlying PIR Π. In particular, the adversary may directly use the dual PIR's correctness to distinguish transcripts produced from Q and those transcripts produced using Q'.

As a note, the privacy adversary to the underlying PIR Π only has access to a crypto oracle O (used by Π to generate the challenge transcript) whereas the dual PIR requires both a crypto oracle O and a random oracle RO. As Π only uses O and does not ever access RO, we will have our adversary lazily sample the random oracle RO locally that will be used by the dual PIR construction.

Lemma 4.9. *Suppose the underlying PIR Π is ε-correct and δ-private. Furthermore, we consider our dual PIR construction $\Pi' = (\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}, \mathsf{Hint}^{\mathsf{O},\mathsf{RO}}, \mathsf{Recon}^{\mathsf{O},\mathsf{RO}})$ for any fixed query sequence Q'. For any query sequence $Q \neq Q'$ and database D, consider an execution of the dual PIR to obtain $(\boldsymbol{\tau}, \mathbf{S}) \leftarrow \mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$, $\mathbf{c} \leftarrow \mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q, (\boldsymbol{\tau}, \mathbf{S}))$ and $\mathbf{ans} \leftarrow \mathsf{Recon}^{\mathsf{O},\mathsf{RO}}(Q, (\boldsymbol{\tau}, \mathbf{S}), \mathbf{c})$. Then,*

$$\Pr[\mathbf{ans} = D[Q]] \geq \varepsilon - 2^{-s+1} - 2^{-n+1} - 2\delta$$

over the random choice of oracles O and RO as well as the internal randomness of the algorithms.

Proof. Towards a contradiction, suppose there exists a query sequence $Q \neq Q'$ for database D such that

$$\Pr[\mathbf{ans} = D[Q]] < \varepsilon - 2^{-s+1} - 2^{-n+1} - 2\delta.$$

We construct the following adversary $\mathcal{A}$ for the privacy game of the underlying PIR Π that has access to the crypto oracle O. The adversary is explicitly defined for $Q_0 = Q'$ and $Q_1 = Q$ as the two query sequences and D as the challenge database.

$\mathcal{A}^{\mathsf{O}}(1^\lambda, \tau, E)$:

1. Parse $\tau = (\tau_{\mathsf{Client}}, \tau_{\mathsf{Server}})$ as the client and server communication respectively.
2. Execute the server-side execution of the query algorithm $\mathsf{Query}^{\mathsf{O}}_{\mathsf{Server}}(\tau_{\mathsf{Client}}; E)$ where we simulate the client using client's communication in the transcript. Record all server performed queries to the crypto oracle O as S.
3. Lazily sample a local random oracle RO. For any query $x \in \{0,1\}^*$, check if $\mathsf{RO}(x)$ is defined. If not, set $\mathsf{RO}(x)$ to be a uniformly random string. Then, return $\mathsf{RO}(x)$.
4. Execute $H \leftarrow \mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q, (\tau, S))$, where the step of Hint computing $E \leftarrow \mathsf{Init}(1^\lambda, D)$ using $\omega_{\mathsf{Server}} = \mathsf{RO}(0)$ is omitted and the encoded database E received as input is used in its place.
5. Execute $R \leftarrow \mathsf{Recon}^{\mathsf{O},\mathsf{RO}}(Q, (\tau, S), H)$.
6. If $R = D[Q]$, output 1.
7. If $R \neq D[Q]$, output 0.

First, we claim that the execution of $\mathcal{A}$ in the privacy game with challenge bit b perfectly simulates an execution of the dual PIR in which Advice is parameterized by Q_b while Hint and Recon receive the query sequence Q. To see this, view the dual PIR's random oracle RO as follows: the value $\mathsf{RO}(0)$ is set to the challenger's internal randomness ω_{Server} used by Init, while RO at every other point takes the values lazily sampled by $\mathcal{A}$. This view is consistent since Init makes no oracle queries and, by domain separation, the construction uses the value $\mathsf{RO}(0)$ *only* as the encoding randomness: Hint uses it only to recompute E (the step that $\mathcal{A}$ replaces with the challenge $E = \mathsf{Init}(1^\lambda, D)$ computed with randomness ω_{Server}, which is by definition the same value), and Recon never queries $\mathsf{RO}(0)$ at all. With this view, the challenger's execution is exactly $(\boldsymbol{\tau}, \mathbf{S}) \leftarrow \mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q_b}(D)$: note that $\mathcal{A}$'s step 2 recovers exactly the set $\mathbf{S}$ recorded by Advice, since the server's execution during Query is deterministic given $(\mathsf{O}, E, \tau_{\mathsf{Client}})$. Hence steps 4–5 of $\mathcal{A}$ produce $(\mathbf{H}, \mathbf{R})$ distributed exactly as in the dual PIR execution.

Next, we prove that our adversary $\mathcal{A}$ performs a polynomial number of queries to the crypto oracle O. Note that indeed $\mathcal{A}$ queries crypto oracle O only on queries in S corresponding to crypto oracle queries performed by an honest server during the online query phase: step 2 re-executes the deterministic server, Hint and Recon query O only to compute $\mathsf{O}(S)$, and all executions under resampled oracles O_i are simulated locally using only RO, since the sampling algorithm of Lemma 3.6 is deterministic with oracle access only to RO given the input $\mathsf{O}(S)$. Next, we prove that $\mathcal{A}$ wins the privacy game for the underlying PIR Π. Consider the case when τ is output from executing PIR Π with query sequence $Q_0 = Q'$. We note that $\mathcal{A}$ outputs 0 with probability at least

$$1 - \Pr[\mathbf{ans} = D[Q]] > 1 - \varepsilon + 2^{-s+1} + 2^{-n+1} + 2\delta$$

since $\Pr[\mathbf{ans} = D[Q]] < \varepsilon - 2^{-s+1} - 2^{-n+1} - 2\delta$ by our assumption towards a contradiction. Now, consider the case when τ is output from executing PIR Π with query sequence $Q_1 = Q$. In this case, the adversary is executing the dual PIR where Advice is parameterized by Q and both Hint and Recon receive the same query sequence Q as input. By Lemma 4.8, we know that the dual PIR is correct with probability at least $\varepsilon - 2^{-s+1} - 2^{-n+1}$. Therefore, $\mathcal{A}$ outputs 1 with probability at least

$$\Pr[\mathbf{ans} = D[Q]] \geq \varepsilon - 2^{-s+1} - 2^{-n+1}.$$

Averaging the two cases over the uniform challenge bit, $\mathcal{A}$ outputs the correct bit with probability

$$\frac{1}{2}\left(1 - \Pr[\mathbf{ans} = D[Q] \mid b = 0]\right) + \frac{1}{2}\Pr[\mathbf{ans} = D[Q] \mid b = 1] > \frac{1}{2} + \frac{2\delta}{2} = \frac{1}{2} + \delta,$$

contradicting the δ-privacy of PIR. □

The above essentially shows that the dual PIR construction is correct even if we consider any query sequence $Q \neq Q'$ where Q' is the fixed query sequence parameterized in $\mathsf{Advice}_{Q'}$ of the dual PIR construction. We can now prove our main result as follows:

Proof of Theorem 4.3. First, we note that Lemma 4.9 states that the dual PIR construction is correct when considering any query sequence Q with probability at least $\varepsilon - 2^{-s+1} - 2^{-n+1} - 2\delta$. As this holds for any fixed query sequence Q and fixed database D, we note it also holds over uniformly randomly chosen Q and database D to complete the proof. □

Remark 4.10 (Extending Beyond Oracle-Free Encodings)**.** *Beyond our main claim, our proof technique extends straightforwardly to rule out encodings which use oracles in particular restricted ways:*

- *First, if* $\mathsf{Init}^{\mathsf{O}}$ *makes only* $o(n)$ *queries to its oracle, then we can include these queries along with the transcript* τ *output by* Advice*. This is analogous to the result in Section 5.1.*
- *Second, if the queries made by* $\mathsf{Init}^{\mathsf{O}}$ *are deterministic and do not depend on the database, or can be predicted from the transcript* τ *alone, then* Recon *can compute these queries directly and resample only oracles consistent with them. This is analogous to the result in Section 5.2.*

However, it is unclear how to extend our result to any encoding which depends on its oracle. This is the most natural extension left open by our work.

PIR with Weaker Privacy Guarantees. In our above proof, we note that the adversary performs at most the same number of queries as an honest server. In particular, our adversary makes all crypto oracle queries as performed by an honest server in the underlying PIR during the online Query protocol. Note, the adversary does not need to make any crypto oracle queries corresponding to Init (which makes none for schemes with oracle-free encodings) nor server crypto oracle queries performed during Setup. As a result, we immediately rule out the existence of more efficient weak PIR schemes that only obtain privacy guarantees against adversaries that may perform at most a fixed polynomial number of queries. We point readers to Section 4.6 for more details.

4.4 Impossibility of Dual PIR

In the next lemma, we rule out the existence of dual PIR which are both correct and too efficient. In particular, we show that any dual PIR scheme that supports $k = \Omega(s)$ queries will be correct with probability at most $2^{s-\Omega(k)}$ assuming the upfront t-bit advice is less than half of the database size, $t < n/2$. We will prove this as an information-theoretic result and, thus, it holds even in relative worlds.

Lemma 4.11. *If n, k, t, s are integers with $t < n/2$, n sufficiently large, and $1000 \log n \leq k \leq 0.19n$, then there is no (n, k, t, s)-dual PIR that is ε-correct for $\varepsilon > 2^{s-0.15k}$. Moreover, this is true relative to any oracle O that is independent of the database and queries.*

By setting $k \geq 14s$, we immediately obtain that it is impossible for a dual PIR to be correct with probability 2^{-s}. In other words, there are no interesting constructions of dual PIR when the final s-bit hint string is small compared to the number of queries k such as the setting of $s = o(k)$. Given the prior section showing that any efficient PIR with preprocessing scheme may be transformed into an efficient dual PIR, we immediately can prove our main lower bound.

Proof of Theorem 4.1. Towards a contradiction, suppose there exists a too-efficient-to-be-true PIR with preprocessing scheme with s-bit client storage supporting $k = \Omega(s)$ queries with $k \leq 0.19n$ where the total bits of communication and server cryptographic bit operations is t that obtains ε-correctness and δ-privacy for any choice of $\varepsilon \geq 2^{-s+4}$, $\delta \leq \varepsilon/3$ and $t \leq n/2$. By Theorem 4.3, this implies a $(n, k, t, 3s)$-dual PIR with ε'-correctness where

$$\varepsilon' \geq \varepsilon - 2^{-s+1} - 2^{-n+1} - 2\delta \geq 2^{-s}$$

using the fact that $s \leq n$, which holds as we take $k \geq 27s$ below and $k \leq 0.19n$ (so that $2^{-n+1} \leq 2^{-s+1}$, giving $\varepsilon' \geq \varepsilon/3 - 2^{-s+2} \geq (16/3 - 4) \cdot 2^{-s} \geq 2^{-s}$). Next, we can apply Lemma 4.11 showing that there cannot exist any $(n, k, t, 3s)$-dual PIR that is ε'-correct where $\varepsilon' \geq 2^{-s}$ to achieve a contradiction, which is true whenever $k \geq 27s$ (as the dual PIR's hint size is $3s$, we require $2^{-s} > 2^{3s-0.15k}$, i.e., $0.15k > 4s$). In other words, this implies that it must be that $t > n/2$ meaning that the total bits in the online communication and the server cryptographic bit operations must be at least $\Omega(n)$. This immediately implies that the total online computation is $\Omega(n)$ and the amortized online computation is $\Omega(n/k)$, which is $\Omega(n/s)$ for $k = \Theta(s)$. □

The remainder of this section is proving the impossibility of efficient dual PIR. To prove this impossibility, we will relate the correctness properties of dual PIR with the subkey prediction problem studied in [BKR16] (we also refer readers back to Section 3.3). As a reminder, the subkey prediction considers the probability that an adversary is able to correctly guess a subkey consisting of k random locations of a large n-bit key when given t bits of leakage. We note the collision of parameters is not coincidental as there are strong similarities between the two problems. Interestingly, subkey prediction is used as a tool to prove the security of a construction of symmetric encryption (upper bounding adversarial advantage) in the bounded retrieval model. In contrast, we will use the subkey prediction problem as a way to prove lower bounds and impossibilities about dual PIR constructions (upper bounding dual PIR correctness).

Prior work [BKR16] presented an upper bound for subkey prediction and later use approximations to obtain concrete bounds. However, they omit a proof of asymptotic bounds for the parameters necessary in our proof. We will provide the relevant analysis below using Lemma 3.9

proven in [BKR16]. Before we present the lemma, we recall some notation that will be important. As a reminder, $B_n(r)$ ball of radius r centered at the origin of the n-dimensional boolean hypercube (all points with distance at most r from 0^n. We use $\mathsf{rd}_n(N) := \max\{\hat{r} \in \mathbb{Z}_{\geq 0} : |B_n(\hat{r})| \leq N\}$ as the radius of the largest ball with at most N points.

Lemma 4.12. *For sufficiently large integers n, t, k such that $t \leq n/2$ and $1000 \log n \leq k \leq 0.19n$ and for all (computationally unbounded) adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

$$\Pr\left[\mathcal{A}_2(\mathbf{Q}, \mathcal{A}_1(\mathbf{D})) = \mathbf{D}[\mathbf{Q}]\right] \leq 2^{-0.15k}$$

where $\mathbf{Q}$, the k subkey locations, is uniform over $[n]^k$, $\mathbf{D}$, n-bit large key, is uniform over $\{0,1\}^n$, adversary $\mathcal{A}_1$ outputs at most t bits of leakage, and adversary $\mathcal{A}_2$ outputs k bits.

Proof. We start with the adversarial advantage proven by Bellare, Kane, and Rogaway [BKR16] (see Lemma 3.9) that we present below where we denote $N = 2^{n-t} \geq 2^{n/2}$ and $r = \min(1 + \mathsf{rd}_n(N), n)$:

$$\Pr[\mathcal{A}_2(\mathbf{Q}, \mathcal{A}_1(\mathbf{D})) = \mathbf{D}[\mathbf{Q}]] \leq \frac{1}{N} \cdot \sum_{i=0}^{r} \binom{n}{i} \cdot \left(1 - \frac{i}{n}\right)^k.$$

For convenience, we rewrite this bound using conditional probabilities as follows:

$$\Pr[\mathcal{A}_2(\mathbf{Q}, \mathcal{A}_1(\mathbf{D})) = \mathbf{D}[\mathbf{Q}]] \leq \frac{|B_n(r)|}{N} \cdot \sum_{i=0}^{r} \Pr[|\mathbf{X}| = i \mid \mathbf{X} \in B_n(r)] \cdot \left(1 - \frac{i}{n}\right)^k.$$

for $\mathbf{X}$ uniformly random over $\{0,1\}^n$.

To start, we show that $|B_n(r)| \leq n \cdot N$. First, note that $r \leq \mathsf{rd}_n(N) + 1$. By definition of $\mathsf{rd}_n(N)$, we know that $|B_n(r-1)| \leq N$. Also, $r \geq 2$ since $|B_n(1)| = n + 1 \leq 2^{n/2} \leq N$ for sufficiently large n. Every point of Hamming weight r is obtained from exactly r points of Hamming weight $r-1$ by flipping one of at most n coordinates, so $\binom{n}{r} \leq (n/r) \cdot \binom{n}{r-1} \leq (n/2) \cdot |B_n(r-1)|$. Therefore, we can see that $|B_n(r)| = |B_n(r-1)| + \binom{n}{r} \leq n \cdot |B_n(r-1)| \leq n \cdot N$. As a result, we see that $|B_n(r)|/N \leq n$.

Next, we split the points into two disjoint sets of Hamming weights: those with at most $n/10$ Hamming weight and the remainder with strictly more than $n/10$ Hamming weight. We can rewrite the adversary's guessing probability as:

$$\begin{aligned}
&\Pr[\mathcal{A}_2(Q, \mathcal{A}_1(D)) = D[Q]] \\
&\quad \leq n \cdot \left(\sum_{i=0}^{n/10} \Pr[|x| = i \mid x \in B_n(r)] \cdot \left(1 - \frac{i}{n}\right)^k + \sum_{i=n/10+1}^{r} \Pr[|x| = i \mid x \in B_n(r)] \cdot \left(1 - \frac{i}{n}\right)^k \right) \\
&\quad \leq n \cdot \sum_{i=0}^{n/10} \Pr[|x| = i \mid x \in B_n(r)] \cdot \left(1 - \frac{i}{n}\right)^k + n \cdot \left(\frac{9}{10}\right)^k \\
&\quad \leq n \cdot \sum_{i=0}^{n/10} \Pr[|x| = i \mid x \in B_n(r)] + 2^{-0.151k + \log n} \\
&\quad = n \cdot \Pr[x \in B_n(n/10) \mid x \in B_n(r)] + 2^{-0.15k}.
\end{aligned}$$

where we use the fact that $\log_2(9/10) < -0.152$ and that $k \geq 1000 \log n$.

It remains to bound the first term of the summand. To do this, we will bound the conditional probability that a random element x from $B_n(r)$ has Hamming weight at most $n/10$. That is, we bound the probability $\Pr[\mathbf{X} \in B_n(n/10) \mid \mathbf{X} \in B_n(r)]$. We first re-write this probability as

$$\Pr[\mathbf{X} \in B_n(n/10) \mid \mathbf{X} \in B_n(r)] = \frac{|B_n(n/10)|}{|B_n(r)|}.$$

Next, we show that $|B_n(r)| \geq |B_n(0.11n)|$. To see this, we consider the size of the ball $B_n(0.11n)$ that can be upper bounded as

$$|B_n(0.11n)| = \sum_{i=0}^{0.11n} \binom{n}{i} \leq 2^{h(0.11)n} < 2^{n/2} \leq N$$

using the fact that $N = 2^{n-t} \geq 2^{n/2}$ since $t \leq n/2$. We use the well-known bounds $2^{h(b/a)a}/(a+1) \leq \sum_{i=0}^{b} \binom{a}{i} \leq 2^{h(b/a)a}$ for $b \leq a/2$, where $h(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function. We also use the fact that $h(0.11) < 1/2$ and, thus, $2^{h(0.11)n} < 2^{n/2}$ in the second inequality. This immediately implies that $|B_n(r)| \geq |B_n(0.11n)|$ since $|B_n(r)| \geq N$ since $r = \min(\mathsf{rd}_n(N)+1, n)$ and we note that $\mathsf{rd}_n(N)+1 < n$ since $N \geq 2^{n/2}$. Therefore, we can get that

$$\begin{aligned}\Pr[\mathbf{X} \in B_n(n/10) \mid \mathbf{X} \in B_n(r)] &\leq \frac{|B_n(n/10)|}{|B_n(r)|} \\ &\leq \frac{|B_n(n/10)|}{|B_n(0.11n)|} \leq (n+1)2^{(h(0.1)-h(0.11))n} \leq n2^{-0.03n}.\end{aligned}$$

where again we use the well-known bounds $2^{h(b/a)a}/(a+1) \leq \sum_{i=0}^{b} \binom{a}{i} \leq 2^{h(b/a)a}$ and the fact that $h(0.1) - h(0.11) < -0.0309$, so the last inequality holds for sufficiently large n. Putting it altogether, we obtain the following:

$$\Pr[\mathcal{A}_2(\mathbf{Q}, \mathcal{A}_1(\mathbf{D})) = \mathbf{D}[\mathbf{Q}]] \leq n^2 \cdot 2^{-0.03n} + 2^{-0.151k} \leq 2^{-0.15k}$$

using the fact that $k \leq 0.19n$ and that n is sufficiently large. □

Next, we observe also that any efficient dual PIR construction could equivalently be considered as an extension of constructing a good adversary for subkey prediction. In fact, the subkey prediction game is essentially the same, except that the adversary is not allowed to receive any additional secret function in the last phase of dual PIR. We use this reduction to complete the proof of Lemma 4.11 showing that any dual PIR may be used to construct an adversary in the subkey prediction game. Thus, there does not exist a dual PIR that is both correct and efficient.

Proof of Lemma 4.11. Consider $\mathbf{Q} \xleftarrow{\$} [n]^k$ to be the uniformly random k key locations and $\mathbf{D} \xleftarrow{\$} \{0,1\}^n$ to be the uniformly random n-bit key. We prove the lemma by considering the min-entropy of the random variable $\mathbf{D}[\mathbf{Q}]$ given the outputs of the algorithms $\mathbf{A} \leftarrow \mathsf{Advice}(D)$ and $\mathbf{H} \leftarrow \mathsf{Hint}(\mathbf{D}, \mathbf{Q}, \boldsymbol{\tau})$ in the DPIR correctness game (Definition 4.2). In particular, we will use the fact that

$$\Pr[\mathsf{Recon}(\mathbf{Q}, \mathbf{A}, \mathbf{H}) = \mathbf{D}[\mathbf{Q}]] \leq 2^{-H_\infty(\mathbf{D}[\mathbf{Q}]|\mathbf{Q},\mathbf{A},\mathbf{H})}$$

to prove the bound. So, all that remains is to show that

$$H_\infty(\mathbf{D}[\mathbf{Q}] \mid \mathbf{Q}, \mathbf{A}, \mathbf{H}) \geq \Omega(k) - s.$$

For this, we can first appeal to Lemma 4.12, which shows that

$$H_\infty(\mathbf{D}[\mathbf{Q}] \mid \mathbf{Q}, \mathbf{A}) \geq 0.15k,$$

for $t < n/2$ and n sufficiently large. Finally, we observe that $\mathbf{H}$ is s bits and, thus, supported on at most 2^s elements. This means that $H_\infty(\mathbf{D}[\mathbf{Q}] \mid \mathbf{Q}, \mathbf{A}, \mathbf{H}) \geq 0.15k - s$ via Lemma 3.2, and the claimed lemma follows immediately. □

4.5 Impossibility of (Public-Key) Doubly Efficient PIR

In this section, we prove that public-key (and therefore unkeyed) doubly efficient PIR (DEPIR) with an oracle-free encoding built from blackbox cryptography is impossible. In particular, we show that any such DEPIR constructions requires linear server time per query (which shows that DEPIR is impossible since it is defined to use sublinear time). As a reminder, prior work [LMW25] ruled out the existence of a restricted class of secret-key DEPIR constructions (thus, also public-key/unkeyed DEPIR) in the crypto oracle model. However, their lower bounds required the DEPIR to consist of a single roundtrip query, along with the restriction of a passive server that simply acts as memory without performing any computation. In this work, we rule out the existence of any construction of any public-key DEPIR in the crypto oracle model whose preprocessing (key generation and database encoding) makes no queries to the crypto oracle. Our lower bound makes no assumptions on the online protocol, which may consist of arbitrarily many rounds with an actively computing server making arbitrary use of the crypto oracle. The two results are therefore incomparable: [LMW25] allows oracle-dependent encodings but restricts the query structure, whereas we allow arbitrary query protocols but restrict the preprocessing.

We note that the oracle-free preprocessing restriction is substantive for public-key DEPIR, where known constructions use cryptography heavily inside their encodings (such as the virtual blackbox obfuscation used in [BIPW17]); ruling out blackbox DEPIR with oracle-dependent preprocessing remains an interesting open problem (see also Remark 4.10). For this reason, our result is most interesting for *unkeyed* DEPIR, where the encoding is a public procedure applied to the database. In other words, our result shows that a cryptographic database encoding is necessary: blackbox cryptography used only during the online phase cannot yield sublinear-time PIR.

Note that our syntax allows a PIR to encode an n-bit database D as E. This maps directly onto the *unkeyed* DEPIR syntax. However, we note that our result below additionally rules out *public-key* DEPIR as well. This is because without loss of generality, we can assume that the public-key is included in the encoding E and then sent as the first message of the query algorithm. In more detail, $\mathsf{Init}(1^\lambda, D)$ runs the (oracle-free) key generation and encoding of the public-key DEPIR, outputs E consisting of the public key and encoded database, and discards the secret key, which the online protocol of a public-key DEPIR never uses [BIPW17]. Therefore, any public-key DEPIR with a $o(n)$ size public-key[8] can be transformed into an unkeyed DEPIR with the same parameters at the cost of an additional round and $o(n)$ communication. Since we have no restrictions on the rounds or communication beyond that it is $o(n)$, our bound will apply to both public-key and unkeyed DEPIR.

Improving Theorem 4.1 for DEPIR. The main difference between DEPIR and the PIR with preprocessing setting of Theorem 4.1 is that the Setup algorithm (see Definition 3.7) is a no-op as

[8]Since the public key is generated independently of the database, requiring a $o(n)$ size public key is a very mild condition; still, our results do not rule out public-key DEPIR with public keys of $\Omega(n)$ size.

the client receives no private state. Note, we could try to apply Theorem 4.1 to obtain a lower bound on the query computation immediately. However, this result only applies for $k = \Omega(s)$ queries where $s = \omega(\log(n + \lambda))$. To apply Theorem 4.1, we could pad the public-key DEPIR with a dummy $\omega(\log(n + \lambda))$-bit client storage and obtain that the total query computation over $\omega(\log(n + \lambda))$ queries must be $\Omega(n)$. Instead, we will refine our proof to show that there does not exist any public-key DEPIR that obtains sublinear query computation for even a single query. This proof will still follow the same proof framework of building a dual PIR and then showing that a dual PIR cannot be both correct and efficient.

Lemma 4.13. *Consider any public-key DEPIR construction with an oracle-free encoding for databases of size n where the communication transcript and all server crypto oracle queries during the query algorithm are t bits in total. Furthermore, suppose Π is ε-correct and δ-private. Then, there exists a $(n, 1, t, 0)$-dual PIR with ε'-correctness satisfying $\varepsilon' \geq \varepsilon - 2\delta - 2^{-n+2}$.*

Proof. We build our dual PIR using the public-key DEPIR construction in a blackbox manner following Theorem 4.3 with two modifications. First, the Hint algorithm is modified to perform no operations and always output nothing. Secondly, the Recon algorithm is modified such that it replaces the counter returned by Hint with $c = 0$. The resulting dual PIR has the same correctness analysis with the following differences. Since $s = 0$, the client state is empty, so the uniformly sampled state always equals the honestly sampled one: every transcript match is an acceptance, and Recon (using $c = 0$) outputs the answer of the first matching iteration. Hence the counter-overflow failure of Lemma 4.6 disappears, and the remaining failure modes are the resampling failures (probability 2^{-n} as before) and the event that no match occurs among the $z = 2^{3n}$ iterations. For the latter, rerunning the computation of Lemma 4.5 with threshold 2^{-2n} shows that $p \geq 2^{-2n}$ except with probability $2^{-2n} \cdot 2^{o(n)} \leq 2^{-n}$; conditioned on $p \geq 2^{-2n}$, no match occurs within the $z = 2^{3n}$ iterations with probability at most $(1 - 2^{-2n})^{2^{3n}} \leq e^{-2^n}$. Altogether, the failure probability is at most $2^{-n+1} + e^{-2^n} \leq 2^{-n+2}$. □

Next, we prove a more direct impossibility for the dual PIR construction for a single query $k = 1$ with no additional encoded information after receiving the query since $s = 0$. This is actually equivalent to the well-known one-way communication complexity of the INDEX problem, which was shown to require $\Omega(n)$ communication originally by Kremer et al. [KNR95, KNR99]. For clarity and completeness, we recount a direct proof below with explicit constants using commonly known information theory measures and inequalities (see, [CT06] for the relevant definitions details).

Lemma 4.14 (Due to [KNR99])**.** *For any $t < n/100$, then there is no $(n, 1, t, 0)$-dual PIR that is ε-correct for $\varepsilon \geq 0.6$. Moreover, this is true relative to any oracle O that is independent of the database and queries.*

Proof. Let $\mathbf{A}$ be the t bit output of $\mathsf{Advice}(\mathbf{D})$ for a uniformly random n-bit database $\mathbf{D}$. Then the output $\mathbf{R} \leftarrow \mathsf{Recon}(\mathbf{Q}, \mathbf{A})$ over the choice of a uniformly random index $\mathbf{Q} \in [n]$ (we drop the notation of Hint and its output, since we care about 0-bit outputs).

Now, we consider the success probability of any dual PIR subject to our constraints, which, expanding with the law of total probability over all values $\mathbf{Q}$ can take is

$$\Pr\left[\mathbf{R} = \mathbf{D}[\mathbf{Q}]\right] = \frac{1}{2} + \frac{1}{n}\sum_{Q=1}^{n} \mathbb{E}_{\mathbf{A}}\left[\max_{b \in \{0,1\}} \Pr[\mathbf{D}[Q] = b \mid \mathbf{A}] - \tfrac{1}{2}\right].$$

The quantity in the expectation is exactly $\Delta(\mathbf{D}[Q], \mathbf{b} \mid \mathbf{A})$, the statistical distance of $(\mathbf{D}[Q] \mid \mathbf{A})$ and a uniformly random bit $\mathbf{b}$. Below, we use $D_{\mathsf{KL}}(\mathbf{D}[Q], \mathbf{b} \mid \mathbf{A})$ denotes the KL divergence between the conditional distribution of $\mathbf{D}[Q]$ given $\mathbf{A}$ and the uniform distribution on $\{0,1\}$. Applying Pinsker's inequality pointwise and then Jensen's inequality (using concavity of $\sqrt{\cdot}$) yields

$$\mathbb{E}_{\mathbf{A}}[\Delta(\mathbf{D}[Q], \mathbf{b} \mid \mathbf{A})] \leq \mathbb{E}_{\mathbf{A}}\left[\sqrt{\frac{D_{\mathsf{KL}}(\mathbf{D}[Q] \mid \mathbf{A})}{2}}\right] \leq \sqrt{\frac{\mathbb{E}_{\mathbf{A}}[D_{\mathsf{KL}}(\mathbf{D}[Q] \mid \mathbf{A})]}{2}} = \sqrt{\frac{I(\mathbf{D}[Q]\,;\,\mathbf{A})}{2}},$$

where the last equality uses the fact that $\mathbf{D}[Q]$ is uniform and the definition of mutual information. Next, since each $\mathbf{D}[Q]$ are independent and $|\mathbf{A}| = t$, the chain rule gives $\sum_Q I(\mathbf{D}[Q]; \mathbf{A}) \leq I(\mathbf{D}; \mathbf{A}) \leq t \ln 2$. Finally, we can apply Cauchy–Schwarz, which shows that $\frac{1}{n}\sum_{Q=1}^{n}\sqrt{I(\mathbf{D}[Q]; \mathbf{A})/2} \leq \sqrt{\sum_Q I(\mathbf{D}[Q]; \mathbf{A})/(2n)} \leq \sqrt{t \ln 2/(2n)}$. This finally gives us that

$$\Pr\left[\mathbf{R} = \mathbf{D}[\mathbf{Q}]\right] \leq \frac{1}{2} + C\sqrt{\frac{t}{n}},$$

with explicit constant $C = \sqrt{(\ln 2)/2} \approx 0.589$.

Finally, taking $t < n/100$, we see that no dual PIR can exist that is correct with probability better than $\varepsilon = \frac{1}{2} + \frac{1}{10}\sqrt{\frac{\ln 2}{2}} \approx 0.559$. □

Theorem 4.15. *Relative to any crypto oracle* O*, there does not exist a single-server public-key doubly efficient PIR with an oracle-free encoding (Definition 3.8) that is* ε*-correct and* δ*-private with sublinear query computation* $t = o(n)$ *against computationally unbounded adversaries making* $\mathrm{poly}(\lambda)$ *queries to* O *for any choice of* $\varepsilon \geq 9/10$ *and* $\delta \leq 1/20$*.*

Proof. First, we apply Lemma 4.13 to obtain a $(n, 1, t, 0)$-dual PIR that is ε'-correct where $\varepsilon' \geq \varepsilon - 2\delta - 2^{-n+2} \geq 9/10 - 1/10 - 2^{-n+2} > 3/4$ for sufficiently large n. This immediately contradicts Lemma 4.14 to show that such a public-key DEPIR is impossible. □

4.6 Impossibility of Weak PIR (with Preprocessing)

In this section, we consider the notion of a weak PIR where the privacy guarantees only hold for weaker adversaries that are limited to a fixed polynomial number of queries to the crypto oracle. We note such weaker cryptographic primitives have been studied in other contexts. The most famous example are Merkle puzzles [Mer78] that enable building weak key exchange where honest parties perform q random oracle queries and provides privacy guarantees against any adversary that performs $o(q^2)$ random oracle queries as shown in [BGI08].

To our knowledge, prior works have not ruled out the existence of weak single-server PIR with sublinear computation even without any preprocessing. For example, both of the previous results ruling out sublinear communication single-server PIR [DMO00] and sublinear public-key operations [DH24] reduce to the impossibility of oblivious transfer in the random oracle model. However, replicating the same reductions in [DMO00, DH24] would not rule out weak PIR as there do exist weak oblivious transfer schemes using one-way functions [BGI08].

We show that our prior lower bounds immediately rule out the existence of such weak PIR built from any blackbox cryptography. In particular, our reduction constructs a weak adversary that performs crypto oracle queries that is at most the number of crypto oracle queries performed

by an honest server during the online query protocol of a proper execution of the PIR (see the adversary in the proof of Lemma 4.9). Therefore, our lower bounds immediately apply to weak PIR where the honest parties make q crypto oracles that are secure against any adversaries that perform $O(q)$ crypto oracle queries.

Theorem 4.16. *Relative to any crypto oracle* O*, there does not exist a single-server weak PIR with an oracle-free encoding (which is trivially satisfied by any PIR whose server stores the database unmodified) that is* ε*-correct and* δ*-private with sublinear communication, sublinear server cryptographic operations such that honest parties make at most* q *queries to* O *against computationally unbounded adversaries making* $O(q)$ *queries to* O *for any choice of* $\varepsilon \geq 9/10$ *and* $\delta \leq 1/20$*.*

Proof. This immediately follows from Theorem 4.15 where we use the fact that the adversary constructed in Lemma 4.9 makes at most $O(q)$ crypto oracle queries. □

Note, the above applies to even public-key DEPIR where the server is able to perform arbitrary oracle-free encoding of the database. Note, we can also rule out the existence of single-server weak PIR with preprocessing by applying the same observation to Theorem 4.1.

Theorem 4.17. *Relative to any crypto oracle* O*, there does not exist a single-server PIR with preprocessing scheme with an oracle-free encoding that is* ε*-correct and* δ*-private with* s*-bit client storage and total communication and server cryptographic operations of* $t = o(n)$ *bits supporting* $k = \Theta(s)$ *queries such that honest parties make at most* q *queries to* O *against computationally unbounded adversaries making* $O(q)$ *queries to* O *for any choice of* $s = \omega(\log(n + \lambda))$*,* $\varepsilon = \Omega(2^{-s})$ *and* $\delta \leq \varepsilon/3$*.*

5 Communication Lower Bounds

This section presents some notable corollaries and variations of our main result that yield *communication* lower bounds. In contrast to our computation bound, each of the variations applies to a somewhat restricted class of PIR with preprocessing. As discussed in Section 4 (and we iterate here), there are good reasons to believe these communication bounds cannot apply in full generality.

5.1 PIR with Sublinear Server Cryptographic Operations

This communication lower bound is a direct corollary of Theorem 4.1. Consider a correct PIR protocol with an oracle-free encoding with a batch size $k \geq \Omega(s)$ that requires that a server only makes $o(n)$ total queries to its crypto oracle during the online query protocol. Then, our main theorem shows that this would imply a dual PIR with $o(n)$ advice and a $O(s)$-bit hint that contradicts information-theoretic bounds. We state this formally below.

Corollary 5.1. *Relative to any crypto oracle* O*, any single-server PIR with preprocessing scheme with an oracle-free encoding (Definition 3.8) that is* ε*-correct and* δ*-private with* s*-bit client storage and total of* $t = o(n)$ *server-side queries to* O*, supporting* $k = \Omega(s)$ *queries with* $k \leq 0.19n$ *against computationally unbounded adversaries making* $\mathrm{poly}(\lambda)$ *queries to* O*, requires total communication* $\Omega(n)$*. This holds for any choice of* $s = \omega(\log(n + \lambda))$*,* $\varepsilon = \Omega(2^{-s})$ *and* $\delta \leq \varepsilon/3$*.*

We find it important to consider this lower bound separately from the main result, since it illustrates the limitations of blackbox constructed PIR when the server doesn't use a significant amount of cryptography. Notice that this bound applies to PIR which run in $\Omega(n)$ time but which do not use the oracle more than $o(n)$ times. And in fact, many of the OWF-based PIR with preprocessing require the server issue no OWF evaluations, e.g., see [HPPY25]. This bound shows that these approaches are at their communication limit, when supporting an unbounded number of queries. This is because they use $O(n)$ communication across $k = \Theta(s)$ queries to refresh a client's state, which is optimal given the above corollary.

Perhaps more importantly, this corollary shows significant limitations of what can be achieved by combining server-side and client-side preprocessing (since our definition allows database encoding). It has been an open question, whether one could decrease the (amortized) communication of OWF-based, single-server PIR by combining server and client preprocessing. The above shows that this is not possible from blackbox cryptography without the server using at least $\Omega(n)$ calls to its crypto oracle (which is achievable via FHE). However, it remains open whether one could improve communication using $\Omega(n)$ server-side evaluations of a weaker oracle like a random oracle, which may be more practically efficient than the FHE-based construction.

5.2 PIR with Communication-Determined Server Cryptographic Operations

Our next communication lower bound follows from similar techniques used in our proof of Theorem 4.1. We observe that for a specific class of PIR with preprocessing, we can actually simplify the PIR to dual PIR reduction. In particular, we define the following class of PIR which are "communication-determined," meaning that the server-side queries to O depend only on the transcript τ and not on the database itself. More formally, we provide the following definition.

Definition 5.2. *A PIR protocol* $\Pi^{\mathsf{O}} = (\mathsf{Init}^{\mathsf{O}}, \mathsf{Setup}^{\mathsf{O}}, \mathsf{Query}^{\mathsf{O}})$ *is* communication-determined *if the server-side queries on the* $\mathsf{Query}^{\mathsf{O}}$ *protocol are a function of the online transcript* τ *and* O*. In other words, there exists an oracle algorithm* $\mathsf{SvrQ}^{\mathsf{O}}(\tau)$*, issuing* $\mathrm{poly}(\lambda)$ *queries to* O*, which outputs* S*, the* O *queries by the server. (More formally, it must return the server queries with probability 1 for every* Q *and* D *over an honestly sampled transcript* τ *and oracle* O*.)*

Next, we present the dual PIR used for the following theorem. We highlight the differences from the version used in Theorem 4.3 with old code in gray and new code in blue. We then sketch the proof of the reduction theorem and state the corresponding corollary. For brevity, we leave out the intermediate lemmas as they are nearly identical to those used to prove Theorem 4.3 with the exception that the server side queries are computed by each party. Roughly speaking, the old difference is that the set S of server-side queries is computed by each party instead of being sent along with the transcript.[9]

1. $\mathsf{Advice}^{\mathsf{O},\mathsf{RO}}_{Q'}(D)$:
   - Run $E \leftarrow \mathsf{Init}(1^\lambda, D)$ using $\omega_{\mathsf{Server}} = \mathsf{RO}(0)$ as the algorithm's randomness.
   - Run $(\mathsf{st}; \perp) \leftarrow \mathsf{Setup}^{\mathsf{O}}(1^\lambda; E)$.

[9]For our reduction, we could allow that SvrQ takes Q as input, but this is essentially equivalent to the given definition. If the output depended on Q in a (computationally) noticeable way, then a server would be able to break the privacy of PIR.

- Run $(\mathsf{ans}; \perp \mid \tau) \leftarrow \mathsf{Query}^{\mathsf{O}}(\mathsf{st}, Q'; E)$.
- Record all server-side oracle O queries executed during the Query algorithm as S.
- Output $A \leftarrow \tau$. Output $A \leftarrow (\tau, S)$.

2. $\mathsf{Hint}^{\mathsf{O},\mathsf{RO}}(D, Q, A)$:
   - Parse $A = \tau$. Parse $A = (\tau, S)$.
   - Compute $S \leftarrow \mathsf{SvrQ}^{\mathsf{O}}(\tau)$.
   - Compute $\mathsf{O}(S) \leftarrow \{(q, \mathsf{O}(q)) \mid q \in S\}$.
   - Run $E \leftarrow \mathsf{Init}(1^\lambda, D)$ using $\omega_{\mathsf{Server}} = \mathsf{RO}(0)$ as the algorithm's randomness.
   - Set $z \leftarrow 2^{3n+2s}$.
   - Construct virtual random oracles $\mathsf{RO}_0, \mathsf{RO}_1, \ldots, \mathsf{RO}_z$ such that $\mathsf{RO}_i(\cdot) = \mathsf{RO}(i \mid\mid \cdot)$ and $i \in \{0, 1, \ldots, z\}$ is represented using a string of $O(n+s)$ bit length.
   - Instantiate counter $c \leftarrow 0$.
   - For $i \in \{1, \ldots, z\}$ and while $c < 2^{3s}$:
     (a) Sample oracle $\mathsf{O}_i \leftarrow \mathcal{A}^{\mathsf{RO}_i}(\mathsf{O}(S))$ such that $\mathsf{O}_i(S) = \mathsf{O}(S)$ using Lemma 3.6 with parameter $\zeta = 6n$. If the algorithm fails to sample crypto oracle O_i, then output $H \leftarrow \perp$.
     (b) Compute $(\mathsf{st}_i, \omega_{\mathsf{Client},i}) \leftarrow \mathsf{RO}_0(i)$.
     (c) Compute $(\mathsf{st}'_i; \perp) \leftarrow \mathsf{Setup}^{\mathsf{O}_i}(1^\lambda; E)$.
     (d) Compute $(\mathsf{ans}; \perp \mid \tau_i) \leftarrow \mathsf{Query}^{\mathsf{O}_i}(\mathsf{st}_i, Q; E)$ using $\omega_{\mathsf{Client},i}$ as the randomness for the client side of the Query protocol.
     (e) If $\tau_i = \tau$:
       i. If $\mathsf{st}_i = \mathsf{st}'_i$, output $H \leftarrow c$.
       ii. Increment $c \leftarrow c + 1$.
   - Output $H \leftarrow \perp$.

3. $\mathsf{Recon}^{\mathsf{O},\mathsf{RO}}(Q, A, H)$:
   - Parse $A = \tau$. Parse $A = (\tau, S)$.
   - Compute $S \leftarrow \mathsf{SvrQ}^{\mathsf{O}}(\tau)$.
   - Parse $H = c$.
   - Parse $\tau = (\tau_{\mathsf{Client}}, \tau_{\mathsf{Server}})$ as client and server communication respectively.
   - Compute $\mathsf{O}(S) \leftarrow \{(q, \mathsf{O}(q)) \mid q \in S\}$.
   - Set $z \leftarrow 2^{3n+2s}$.
   - Construct virtual random oracles $\mathsf{RO}_0, \mathsf{RO}_1, \ldots, \mathsf{RO}_z$ such that $\mathsf{RO}_i(\cdot) = \mathsf{RO}(i \mid\mid \cdot)$ and $i \in \{0, 1, \ldots, z\}$ is represented using string of $O(n+s)$ bits.
   - Instantiate counter $c' \leftarrow 0$.
   - For $i \in \{1, \ldots, z\}$:

(a) Sample oracle $\mathsf{O}_i \leftarrow \mathcal{A}^{\mathsf{RO}_i}(\mathsf{O}(S))$ such that $\mathsf{O}_i(S) = \mathsf{O}(S)$ using Lemma 3.6 with parameter $\zeta = 6n$. If the algorithm fails to sample crypto oracle O_i, then output $R \leftarrow \perp$.

(b) Compute $(\mathsf{st}, \omega_{\mathsf{Client}}) \leftarrow \mathsf{RO}_0(i)$.

(c) Compute $(\mathsf{ans} \mid \tau'_{\mathsf{Client}}) \leftarrow \mathsf{Query}^{\mathsf{O}_i}_{\mathsf{Client}}(\mathsf{st}, Q; \tau_{\mathsf{Server}})$ using ω_{Client} as the randomness for the client side of the Query protocol.

(d) If $\tau'_{\mathsf{Client}} = \tau_{\mathsf{Client}}$:

i. If $c' = c$, output $R \leftarrow \mathsf{ans}$.

ii. Increment $c' \leftarrow c' + 1$.

– Output $R \leftarrow \perp$.

Theorem 5.3. *Suppose there is a PIR $\Pi = (\mathsf{Init}, \mathsf{Setup}^{\mathsf{O}}, \mathsf{Query}^{\mathsf{O}})$ with an oracle-free encoding (Definition 3.8) supporting batch queries of size k for databases of size n with client state of s bits, and the communication transcript is t bits. Furthermore, suppose Π is ε-correct and δ-private and communication-determined.*

For any fixed query sequence Q', our construction is a $(n, k, t, 3s)$-dual PIR with ε'-correctness satisfying $\varepsilon' \geq \varepsilon - 2\delta - 2^{-s+1} - 2^{-n+1}$.

Proof sketch. The proof of this theorem follows the same line of reasoning as Theorem 4.3 (in particular, the oracle-free encoding assumption is used in the same places; see Claim 4.4). Each of Lemmas 4.5, 4.6, 4.7, 4.8, and 4.9 can be analogously shown for the construction outlined above. Most of these do not change significantly, such as bounding the counter, number of oracles needed, and switching from one query to another. Lemma 4.7 changes the most as we now must argue that the output distribution of the two games are the same where $\mathbf{S}$ is now computed from $\boldsymbol{\tau}$ instead of output by Advice. However, this change adds a single step to the proof to establish that if $\boldsymbol{\tau}$ is distributed identically to $\boldsymbol{\tau}'$ then $(\boldsymbol{\tau}, \mathbf{S})$ is identically distributed to $(\boldsymbol{\tau}', \mathbf{S}')$. This follows from the straightforward observation that $\mathbf{S} \leftarrow \mathsf{SvrQ}^{\mathsf{O}}(\boldsymbol{\tau})$ is deterministic and correct with probability 1. □

Next, we state the corollary, which is most analogous to Theorem 4.1 and follows from the above Theorem 5.3 combined with Lemma 4.11.

Corollary 5.4. *Relative to any crypto oracle O, any single-server PIR with preprocessing scheme with an oracle-free encoding that is ε-correct, δ-private, and communication-determined with s-bit client storage, supporting $k = \Theta(s)$ queries against computationally unbounded adversaries making $\mathrm{poly}(\lambda)$ queries to O, requires total communication $\Omega(n)$. This holds for any choice of $s = \omega(\log(n + \lambda))$, $\varepsilon = \Omega(2^{-s})$ and $\delta \leq \varepsilon/3$.*

Comparison to "database oblivious/independent" PIR. Prior work [ISW24] considered a definition of "database-oblivious" PIR. This is arguably a more restricted definition than Definition 5.2 above though technically incomparable. Database-oblivious PIR requires both that the database is not encoded and that all client messages depend only on their randomness and not on server responses. Our definition makes neither of these restrictions. Instead, we only require that the server's use of cryptography must be fully determined by the transcript itself and therefore cannot depend on the entire database.

In fact, under our definition the server crypto oracles may depend on the database if they appear in the communication transcript (so, they do not need to be completely database-independent). In fact, this lower bound may be generalized to the case where the server crypto oracles depend on a sublinear subset of the database or to a function of the database with output size $o(n)$. To our knowledge, this generality does not widen the applicability of our lower bound as, all known PIR schemes either make the server crypto oracles completely independent of the database (such as those based on OWFs) or the server crypto oracle queries depend on the entire database (such as schemes using FHE).

5.3 PIR with Perfect Privacy

Our next communication lower bound applies to any PIR which is *perfectly private* against all adversaries with $\text{poly}(\lambda)$ queries to a specific crypto oracle O and whose client-side Query algorithm is deterministic given its state, its queries, the incoming server messages, and O (e.g., the state contains a seed sd and the client derives its query randomness as $\mathsf{O}(\mathsf{sd} \mid\mid \cdot)$).[10] Recall that perfect privacy is equivalent to our traditional privacy when all adversaries have advantage $\delta = 0$. This assumption simplifies many of the difficulties that arise in the proof of Theorem 4.1. Despite this simplification, perfect privacy is not as strong of an assumption as one might originally think. In particular, we require perfect privacy only with respect to an *idealized crypto oracle*. For example, recent works, like [HPPY25], are perfectly secret relative to a random oracle, derive their client query randomness from the random oracle and key material in their state, and therefore this bound applies to them. We also note that, in contrast to our previous lower bounds, this result requires no oracle-free encoding restriction on $\mathsf{Init}^{\mathsf{O}}$, as the dual PIR below never resamples crypto oracles.

The simplified dual PIR for this result is defined as follows, where all algorithms have access to the crypto oracle O but share no other randomness.

1. $\mathsf{Advice}^{\mathsf{O}}_{Q'}(D)$:
   - Run $E \leftarrow \mathsf{Init}^{\mathsf{O}}(1^\lambda, D)$.
   - Run $(\mathsf{st}; \perp) \leftarrow \mathsf{Setup}^{\mathsf{O}}(1^\lambda; E)$.
   - Run $(\mathsf{ans}; \perp \mid \tau) \leftarrow \mathsf{Query}^{\mathsf{O}}(\mathsf{st}, Q'; E)$.
   - Parse $\tau = (\tau_{\mathsf{Client}}, \tau_{\mathsf{Server}})$
   - Output $A \leftarrow \tau_{\mathsf{Server}}$.
2. $\mathsf{Hint}^{\mathsf{O}}(D, Q, A)$:
   - Sample $\mathsf{st} \xleftarrow{\$} \{0,1\}^s$
   - Output $H \leftarrow \mathsf{st}$.
3. $\mathsf{Recon}^{\mathsf{O}}(Q, A, H)$:
   - Parse $A = \tau_{\mathsf{Server}}$.

[10] For example, the schemes of [RMS24, HPPY25] derive all query randomness from the oracle and key material in their state. More generally, any scheme can store its query randomness in its state, preserving privacy exactly with a larger state.

- Parse $H = \mathsf{st}$.
- Compute $(\mathsf{ans} \mid \tau'_{\mathsf{Client}}) \leftarrow \mathsf{Query}^{\mathsf{O}}_{\mathsf{Client}}(\mathsf{st}, Q; \tau_{\mathsf{Server}})$, where the client derives its randomness from O and the seed in st.
- Output $R \leftarrow \mathsf{ans}$.

Notably, this dual PIR only requires that the advice protocol output the server-side transcript, meaning that we obtain a stronger lower bound in the sense that it is a lower bound on the *download* communication online, regardless of how much upload communication the PIR uses. This also allows our hint algorithm to avoid searching for consistent states and instead just try and guess the state that was used in the initial phase directly. The following theorem and subsequent corollary remove many of the more technical parts of our main result.

Theorem 5.5. *Suppose there is a PIR* $\Pi = (\mathsf{Init}^{\mathsf{O}}, \mathsf{Setup}^{\mathsf{O}}, \mathsf{Query}^{\mathsf{O}})$ *whose client-side* Query *algorithm is deterministic given its state as above, supporting batch queries of size k for databases of size n with client state of s bits, and the download communication transcript is t bits. Furthermore, suppose* Π *is ε-correct and perfectly private.*

For any fixed query sequence Q', our construction is a (n, k, t, s)-dual PIR with ε'-correctness satisfying $\varepsilon' \geq \varepsilon \cdot 2^{-s}$.

Proof. To show the above dual PIR is correct, we can argue much more directly compared to our previous Theorem 4.1. In particular, we first observe that perfect privacy implies that for any database D and two query sets Q, Q', the probability that the above dual PIR outputs the correct answer $D[Q]$ is identical whether using $\boldsymbol{\tau}$ output from $\mathsf{Advice}^{\mathsf{O}}_{Q'}$ or from $\mathsf{Advice}^{\mathsf{O}}_{Q}$. Indeed, a privacy adversary given (τ, E) can sample a random state $\mathsf{st} \xleftarrow{\$} \{0,1\}^s$, run the deterministic client-side replay $\mathsf{Query}^{\mathsf{O}}_{\mathsf{Client}}(\mathsf{st}, Q; \tau_{\mathsf{Server}})$ itself, and check whether the output equals $D[Q]$; this adversary makes $\mathrm{poly}(\lambda)$ queries to O, its execution with challenge bit b is exactly the dual PIR with Advice parameterized by Q_b, and applying $\delta = 0$ to it and its complement forces the two probabilities to be equal. (This is a simplification of the argument given in the proof of Lemma 4.9.) In particular, we consider the following two games:

1. $\mathbf{Game}_0$: The dual PIR correctness game.
2. $\mathbf{Game}_1$: The dual PIR correctness game replacing $\mathsf{Advice}^{\mathsf{O}}_{Q'}(\mathbf{D})$ with $\mathsf{Advice}^{\mathsf{O}}_{\mathbf{Q}}(\mathbf{D})$.

Now, consider that the $\mathsf{ans}_{\mathsf{Advice}}$ computed in $\mathsf{Advice}^{\mathsf{O}}_{Q}$ is computed just as in the regular PIR game and therefore the probability that it is equal to $D[Q]$ is at least ε by assumption. Further, notice that $\mathsf{Query}^{\mathsf{O}}_{\mathsf{Client}}$ is deterministic given the state st (which contains the seed determining the client's randomness), queries Q, oracle O, and transcript τ_{Server}. Further, conditioned on the event that the state sampled in Hint, $\mathsf{st}_{\mathsf{Hint}} = \mathsf{st}_{\mathsf{Advice}}$, the output of the Recon, $\mathsf{ans}_{\mathsf{Recon}} = \mathsf{ans}_{\mathsf{Advice}}$. Let p_i denote the probability that the event

$$\mathsf{Recon}^{\mathsf{O}}(\mathbf{Q}, \mathbf{A}, \mathbf{H}) = \mathbf{D}[\mathbf{Q}]$$

occurs in $\mathbf{Game}_i$, then we have

$$p_0 = p_1 \geq \Pr[\mathsf{st}_{\mathsf{Hint}} = \mathsf{st}_{\mathsf{Advice}}] \cdot \Pr[\mathsf{ans}_{\mathsf{Recon}} = \mathbf{D}[\mathbf{Q}] \mid \mathsf{st}_{\mathsf{Hint}} = \mathsf{st}_{\mathsf{Advice}}] \geq \varepsilon \cdot 2^{-s},$$

where the probabilities on the right are taken in $\mathbf{Game}_1$, which establishes the claim that the proposed dual PIR is correct with probability $\varepsilon' \geq \varepsilon \cdot 2^{-s}$. □

We note that the proof degrades gracefully for δ-private schemes: the adversary above then has advantage $|p_0 - p_1|/2 \leq \delta$, yielding a dual PIR with $\varepsilon' \geq \varepsilon \cdot 2^{-s} - 2\delta$. Hence, the corollary below also holds for δ-private schemes with $\delta \leq \varepsilon \cdot 2^{-s-2}$, though this tolerance is necessarily exponentially small.

Next, we state the corollary, which is most analogous to Theorem 4.1 and follows from the above Theorem 5.5 combined with Lemma 4.11.

Corollary 5.6. *Relative to any crypto oracle* O*, any single-server PIR with preprocessing scheme with a deterministic client-side* Query *algorithm as above that is* ε*-correct and perfectly private, with* s*-bit client storage, supporting* $k = \Omega(s)$ *queries against computationally unbounded adversaries making* $\mathrm{poly}(\lambda)$ *queries to* O*, requires total download communication* $\Omega(n)$*. This holds for any choice of* $s = \omega(\log(n + \lambda))$ *and* $\varepsilon = \Omega(2^{-s})$*.*

6 Symmetric PIR without Online Public-Key Cryptography

In this section, we study symmetric PIR (SPIR) with client-side preprocessing focused on online efficiency, which to our knowledge has only been studied in one concurrent work [LWZW25].[11] We begin with an overview of how this direction relates to prior works on SPIR, and we extend our lower bound to this setting. We then give both simple and complex constructions of SPIR with no online public-key operations, the best of which match the online communication and time of the best PIR constructions.

Recall that SPIR is equivalently formulated as efficient 1-out-of-n Oblivious Transfer (OT). And, it is well known from [IR89] that OT cannot be built from one-way functions in a black-box way. Regardless of the efficiency limitations, this separation also applies to any SPIR with preprocessing as well, since we know that across the whole protocol an OT has been performed. This seemingly makes it impossible to base SPIR on one-way functions.

However, there is still an asymmetry between the offline and online phases of a PIR/SPIR protocol. And, since using public-key operations is generally more expensive than symmetric-key ones (like OWFs), we would ideally only rely on heavy computation in the offline phase. This is a problem that has been theoretically and practically considered in the context of OT-extensions, which allow two parties to run a few OTs offline and then use symmetric-key primitives online to quickly perform even more OTs [Bea96, IKNP03, ALSZ13, KK13, ALSZ15].

6.1 Formal model

The primary difference from other versions of secure two-party computation (2PC) with preprocessing is that we wish to build SPIR with *no per-client storage* and *no online public-key operations*.

Shared offline/online server randomness. In our SPIR syntax, the server obtains a random string R that is the same between the offline and online phases. As we show later in Section 6.2, this shared randomness is necessary. In particular, any perfectly private SPIR with preprocessing supporting k queries must allow the server to maintain at least k-bits of shared randomness between

[11] The work of [LWZW25] requires two servers. Moving our main construction's offline computation to a second server can match their construction's parameters up to polylog factors, but will only obtain computational rather than statistical security.

the offline and online phases.[12] If we allow randomized encodings, we could have instead fit this into an Init procedure, but for conceptual simplicity, we remove the Init function. Fortunately, our lower bounds apply to any PIR whose encoding procedure makes no crypto oracle queries, and the encoding induced by a SPIR (the database D stored together with the plain uniform randomness R) requires no oracle access. Hence, our lower bounds will similarly apply to any SPIR with preprocessing.

Per-client storage. Beyond a shared random string or encoding, we could allow the server to store its own state from interacting with the client in the offline phase. However, in a setting with many clients this would be impractical as the server storage grows with the size of the user base. Additionally, if both the client and server are allowed a state, then the (computational version) of the problem is exactly efficient oblivious transfer with preprocessing (or 2PC with preprocessing more generally), which has been studied in prior works [CDG+17, DG17, DKL+23]. But, those works do not capture the same asymmetric preprocessing between the sender and receiver that is present between the client and server in SPIR.

Adversarial model. We only prove our constructions secure against semi-honest servers and clients. We make this simplifying assumption because our schemes are primarily proofs of concept, relying heavily on generic 2PC. Using some generic 2PC which is secure against malicious adversaries, we could have maliciously secure SPIR with client-side preprocessing (for at least our toy constructions); however, there are known barriers for generic maliciously-secure 2PC (see [GMW87, CKL03, LP07, HV16] and the references therein). Therefore, we leave maliciously-secure SPIR with preprocessing as an interesting open problem for future work.

Definition 6.1 (Symmetric Private Information Retrieval)**.** *A k-batch* symmetric private information retrieval with preprocessing scheme (SPIR) *is a pair of efficient two-party protocols $\Pi = (\mathsf{Setup}, \mathsf{Query})$ parameterized by a database size n, batch size k, download t, state size s, and randomness r with the following syntax:*[13]

- $\mathsf{Setup}(1^\lambda; 1^\lambda, D, R) \to (\mathsf{st}; \perp \mid \tau_{\text{off}})$*, where the client and server receive as input the security parameter λ, and the server additionally receives a database $D \in \{0,1\}^n$ and private randomness $R \in \{0,1\}^r$. At the end of the protocol, the client outputs a state $\mathsf{st} \in \{0,1\}^s$, the server receives no output, and τ_{off} is the transcript of the interaction.*

- $\mathsf{Query}(\mathsf{st}, Q; D, R) \to (\mathsf{ans}; \perp \mid \tau_{\text{on}})$*, where the client takes its precomputed state* st *and an ordered set of queries $Q \in [n]^k$, and the server receives a database D and private randomness $R \in \{0,1\}^r$. At the end of the protocol, the client outputs* $\mathsf{ans} \in \{0,1\}^k$ *and the server outputs nothing. We denote the client messages $\tau_u \in \{0,1\}^*$ and the server messages $\tau_d \in \{0,1\}^t$, so the transcript of this entire protocol is $\tau_{\text{on}} = (\tau_u, \tau_d)$.*

A SPIR is ε-correct *if for all sufficiently large λ, tuples $Q \in [n]^k$, and $D \in \{0,1\}^n$,*

$$\Pr\left[\mathsf{Query}(\mathbf{st}, Q; D, \mathbf{R}) = (D[Q]; \perp) \;\middle|\; \begin{array}{c} \mathbf{R} \xleftarrow{\$} \{0,1\}^r \\ (\mathbf{st}; \perp \mid \tau_{\text{off}}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda; 1^\lambda, D, \mathbf{R}) \end{array}\right] \geq \varepsilon.$$

[12] For computational privacy, the servers can share $O(\lambda)$ bits of randomness which can be expanded with a pseudorandom generator.

[13] Note that k, t, s, r may be functions of λ and n; however, we omit this notation for simplicity.

A SPIR is δ-private *if for all sufficiently large* λ*, tuples* $Q_0, Q_1 \in [n]^k$*, efficient adversaries* $\mathcal{A}$*, and* $D \in \{0,1\}^n$*,*

$$\Pr\left[\mathcal{A}(1^\lambda, D, \mathbf{R}, \boldsymbol{\tau}_{\text{off}}, \boldsymbol{\tau}_{\text{on}}) = b \;\middle|\; \begin{array}{c} \mathbf{R} \xleftarrow{\$} \{0,1\}^r \; ; \; b \xleftarrow{\$} \{0,1\} \\ (\mathbf{st}; \perp | \boldsymbol{\tau}_{\text{off}}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda; 1^\lambda, D, \mathbf{R}) \\ (\mathbf{ans}; \perp | \boldsymbol{\tau}_{\text{on}}) \xleftarrow{\$} \mathsf{Query}(\mathbf{st}, Q_b; D, \mathbf{R}) \end{array}\right] \leq \frac{1}{2} + \delta.$$

A SPIR is γ-data private *if for all sufficiently large* λ*, tuples* $Q \in [n]^k$*, databases* $D_0, D_1 \in \{0,1\}^n$ *with* $D_0[Q] = D_1[Q]$*, efficient adversaries* $\mathcal{A}$*,*

$$\Pr\left[\mathcal{A}(1^\lambda, Q, \mathbf{st}, \boldsymbol{\tau}_{\text{off}}, \boldsymbol{\tau}_{\text{on}}) = b \;\middle|\; \begin{array}{c} \mathbf{R} \xleftarrow{\$} \{0,1\}^r \; ; \; b \xleftarrow{\$} \{0,1\} \\ (\mathbf{st}; \perp | \boldsymbol{\tau}_{\text{off}}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda; 1^\lambda, D_b, \mathbf{R}) \\ (\mathbf{ans}; \perp | \boldsymbol{\tau}_{\text{on}}) \xleftarrow{\$} \mathsf{Query}(\mathbf{st}, Q; D_b, \mathbf{R}) \end{array}\right] \leq \frac{1}{2} + \gamma.$$

One may also consider SPIR relative to an oracle O*, as was done in Definition 3.7. We omit this notation to avoid redundancy. Notably, security in such a model only requires that adversaries* $\mathcal{A}$ *submit only polynomially-many queries to* O*.*

A SPIR scheme is called perfectly *correct, private, and data private when* $\varepsilon = 1$*,* $\delta = 0$*, and* $\gamma = 0$*, respectively.*

6.2 Limitations of SPIR without Public-Key Cryptography

In this section, we briefly discuss limitations of SPIR with preprocessing. Obviously, the bounds in previous sections will still apply to SPIR with preprocessing, since they apply to any PIR with an oracle-free encoding procedure and the encoding induced by a SPIR similarly makes no oracle queries (i.e., sharing plain randomness for the server doesn't circumvent our lower bounds).

Beyond these lower bounds, however, the introduction of data privacy allows us to prove some limitations on the capability of SPIR with no online public-key operations. First, we show that any scheme with s-bit client storage may support at most $k \leq s$ queries in the random oracle model. Secondly, we show that SPIR supporting k queries must allow the server at least k-bits of shared randomness between the offline and online phases. These are presented in the context of perfectly private and data private SPIR to simplify the proofs as they are meant just to better explain the limitations of our own construction.

Limits on Number of Queries. Here, we show that one can use data privacy to simplify and potentially improve over our general lower bounds in previous sections. In particular, we show a simple proof that perfectly private and data private SPIR with s bits of preprocessing can only correctly query $k \leq s$. Specifically, Theorem 6.2 below shows that there is no SPIR relative to a random oracle RO which is perfectly private and data private against adversaries with $\text{poly}(\lambda)$ queries to RO, correct, and supports $k > s$ queries — no matter how much communication is allowed.

Unfortunately, this result shows that it is impossible to construct SPIR with preprocessing for an unbounded number of queries without introducing online public-key operations and using blackbox OWFs. Even if one were to allow $\Omega(n)$ communication! This is a key difference from the PIR setting, where there are constructions which can support an unbounded number of queries by using $\Omega(n)$ communication to continuously "refresh" hints without public-key operations. In our main construction, we show how to actually match this bound up to $\text{polylog}(\lambda, n)$ factors with sublinear per-query computation.

Theorem 6.2 (Batch SPIR limitations)**.** *Let $s < k \leq n$ and $\varepsilon > 2^{s-k}$. In the random oracle model, there does not exist a k-batch SPIR scheme with s-bit client storage that is ε-correct, perfectly private, and perfectly data private.*

Proof. We could prove this result using our dual PIR paradigm, but in fact, we can provide the proof more directly. From perfect privacy and data privacy, we know that the messages between the client and server in the online phase are independent of the indices queried Q and of the database D. This means any party can sample a random variable τ_{on} from this distribution for a specific random oracle RO without knowledge of either Q nor D.

This gives a natural compression scheme for k bits into s bits given some common random string, which we interpret as a random oracle RO, client query randomness and a sample of τ_{on} (for RO, without knowledge of D nor Q). Specifically, a sender can set the bits of an otherwise random database D at a canonical set Q. Then, the sender will sample some state st such that $\mathsf{Query}^{\mathsf{RO}}(\mathsf{st}, Q) = D[Q]$ when using the transcript τ_{on} for the server's replies. Finally, they send st (s bits) to the other party. If no such state exists, then fail.

Then, the receiver will run $\mathsf{Query}^{\mathsf{RO}}(\mathsf{st}, Q)$ themselves to recover the k bits. Here we assume that the client side of Query runs with the same random coin tosses as the sender, which can be done without loss of generality by putting the client's randomness in the common random string. Notice that, if the underlying SPIR scheme is correct with probability ε, then this scheme will also be correct with probability at least ε (over the choice of client randomness, τ_{on}, and RO).

Note there are at most 2^s possible outputs of the first phase. For any fixed randomness, the receiver can only output at most 2^s strings, so at most 2^s of the 2^k possible values of $D[Q]$ are recovered correctly. Since correctness must hold for every database D, averaging over all 2^k values of $D[Q]$ shows that the scheme is correct with probability at most $2^s/2^k = 2^{s-k}$. □

Necessity of Shared Randomness. We observe that any SPIR with client-side preprocessing can be transformed into a (non-simultaneous) 2-server SPIR by performing the preprocessing at one server and the only queries at the other. Furthermore, in [GIKM98], a short argument is given that any multi-server SPIR system, with information-theoretic data privacy, must have shared randomness. We review and strengthen this argument to show that any 2-server SPIR requires some minimum amount of randomness, with an argument that fully relativizes. Therefore, any SPIR with preprocessing construction necessitates some amount of shared randomness to avoid public-key operations in the online phase. We start by extending the argument in [GIKM98] as follows:

Lemma 6.3. *There is no 2-server SPIR (without preprocessing) which is perfectly correct, private, and data private and supports $k < n$ queries where the random string $\mathbf{R}$ shared by the two servers (beyond the database D) has entropy $H(\mathbf{R}) < k$.*

Proof. Fix any set Q of k queries, let $\mathbf{D}$ be a uniformly random database, let $\boldsymbol{\omega}$ denote the client's private randomness, and let $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ denote the transcripts with the first and second server, respectively. Each server's messages are a function of its own transcript so far, $\mathbf{D}$, $\mathbf{R}$, and its private randomness, while each client message is a function of $\boldsymbol{\omega}$ and the messages the client has received so far. The argument below holds even relative to an independent oracle O.

First, we claim that each $\boldsymbol{\tau}_i$ is independent of $\mathbf{D}$. By perfect privacy, the distribution of $\boldsymbol{\tau}_i$ does not depend on the queried indices Q and, by perfect data privacy, it depends on $\mathbf{D}$ only through $\mathbf{D}[Q]$. Hence, for any two databases that differ in a single position x, choosing a set of $k < n$

queries that avoids x shows that $\boldsymbol{\tau}_i$ has the same distribution under both databases, and chaining single-bit changes shows that the distribution of $\boldsymbol{\tau}_i$ is the same for every database. Moreover, $\boldsymbol{\tau}_1$ remains independent of $\mathbf{D}$ conditioned on $\boldsymbol{\omega}$: the probability of a transcript $\boldsymbol{\tau}_1$ factors into a client part that depends only on $\boldsymbol{\omega}$ and a server part that depends only on $\mathbf{D}$, $\mathbf{R}$, and the first server's randomness, and the independence above shows that the server part does not depend on $\mathbf{D}$ for any transcript in the support.

Second, by perfect correctness the client recovers $\mathbf{D}[Q]$ from $(\boldsymbol{\omega}, \boldsymbol{\tau}_1, \boldsymbol{\tau}_2)$, and $\mathbf{D}[Q]$ is uniform over $\{0,1\}^k$, so $I(\boldsymbol{\tau}_1\boldsymbol{\tau}_2; \mathbf{D} \mid \boldsymbol{\omega}) \geq H(\mathbf{D}[Q]) = k$. Combining this with the first claim,

$$\begin{aligned} k &\leq I(\boldsymbol{\tau}_1\boldsymbol{\tau}_2; \mathbf{D} \mid \boldsymbol{\omega}) \\ &= I(\boldsymbol{\tau}_1; \mathbf{D} \mid \boldsymbol{\omega}) + I(\boldsymbol{\tau}_2; \mathbf{D} \mid \boldsymbol{\tau}_1, \boldsymbol{\omega}) \\ &= I(\boldsymbol{\tau}_2; \mathbf{D} \mid \boldsymbol{\tau}_1, \boldsymbol{\omega}). \end{aligned}$$

Finally, we show that $I(\boldsymbol{\tau}_2; \mathbf{D} \mid \boldsymbol{\tau}_1, \boldsymbol{\omega}) \leq H(\mathbf{R})$, which completes the proof. Let $\mathbf{m}_2$ and $\mathbf{a}_2$ denote the client's messages and the second server's responses in $\boldsymbol{\tau}_2$. Since $\mathbf{m}_2$ is determined by $(\boldsymbol{\omega}, \boldsymbol{\tau}_1)$ and the earlier responses in $\mathbf{a}_2$, we have $I(\boldsymbol{\tau}_2; \mathbf{D} \mid \boldsymbol{\tau}_1, \boldsymbol{\omega}) = I(\mathbf{a}_2; \mathbf{D} \mid \boldsymbol{\tau}_1, \boldsymbol{\omega})$. We give the argument for a single round of interaction with the second server, so that $\mathbf{m}_2$ is a function of $(\boldsymbol{\omega}, \boldsymbol{\tau}_1)$; the general case follows by applying the same argument to each response in turn using the chain rule. We have

$$\begin{aligned} I(\mathbf{a}_2; \mathbf{D} \mid \boldsymbol{\tau}_1, \boldsymbol{\omega}) &\leq I(\mathbf{a}_2; \mathbf{D}, \boldsymbol{\tau}_1, \boldsymbol{\omega} \mid \mathbf{m}_2) \\ &= I(\mathbf{a}_2; \mathbf{D} \mid \mathbf{m}_2) + I(\mathbf{a}_2; \boldsymbol{\tau}_1, \boldsymbol{\omega} \mid \mathbf{D}, \mathbf{m}_2) \\ &\leq I(\mathbf{a}_2; \mathbf{R} \mid \mathbf{D}, \mathbf{m}_2) \\ &\qquad + I(\mathbf{a}_2; \boldsymbol{\tau}_1, \boldsymbol{\omega} \mid \mathbf{D}, \mathbf{m}_2, \mathbf{R}) \\ &\leq H(\mathbf{R}). \end{aligned}$$

The first inequality uses that $\mathbf{m}_2$ is determined by $(\boldsymbol{\tau}_1, \boldsymbol{\omega})$, and the second uses that $I(\mathbf{a}_2; \mathbf{D} \mid \mathbf{m}_2) = 0$, since the second server's view $(\mathbf{m}_2, \mathbf{a}_2)$ is independent of $\mathbf{D}$ by the first claim. The last inequality uses that $I(\mathbf{a}_2; \boldsymbol{\tau}_1, \boldsymbol{\omega} \mid \mathbf{D}, \mathbf{m}_2, \mathbf{R}) = 0$, because given $\mathbf{m}_2$, $\mathbf{D}$, and $\mathbf{R}$, the response $\mathbf{a}_2$ depends only on the second server's private randomness, which is independent of everything else. Therefore, $H(\mathbf{R}) \geq k$. □

Finally, we prove that this implies shared randomness in the single-server SPIR with preprocessing setting. Roughly speaking, this follows as it is possible to convert any single-server SPIR with preprocessing into a two-server SPIR.

Theorem 6.4. *There is no single-server SPIR with preprocessing which is perfectly correct, private, and data private and supports $k < n$ queries where the server's random string $\mathbf{R}$ shared between the offline and online phases has entropy $H(\mathbf{R}) < k$.*

Proof. We can convert any single-server SPIR with preprocessing into a two-server SPIR as follows. We have one server to execute the offline phase and compute the s-bit hint. We can do this in plaintext since this server will not receive the online query transcript and the preprocessing is done independent of the queried indices. Using the s-bit hint, the client runs the query algorithm with the second server where query privacy holds as the second server does not observe the s-bit hint. For data privacy, we note that the client can only learn one entry per query as the original

single-server SPIR scheme obtained data privacy. The only difference is that the client receives the s-bit hint in plaintext (as opposed to potentially using more complex cryptography such as MPC). Note, the client learns less information about the database in this two-server SPIR scheme as it only learns the s-bit hint output of the offline preprocessing phase. As a result, we obtain a two-server SPIR with the same correctness that is both private and data private while supporting k queries. Finally, we can apply Lemma 6.3 that rules out the existence of such a single-server SPIR. □

6.3 Toy Constructions

Before giving our main construction in Section 6.4, we go over two simple constructions of SPIR using no public-key cryptography in the online phase. For these constructions, we will assume that we have access to some generic secure two-party computation (2PC) [Yao86, GMW87], which can semi-honestly compute the functionality that our Setup protocol calls for. As these constructions are proofs of concept, we leave it to future work to optimize the setup functionality for a specific scheme.

n-bit client storage, $O(k \log n)$-bit communication. Unlike PIR, notice that SPIR is non-trivial even when the client has n bits of storage because the client cannot store the database in plaintext, as this would violate data privacy. However, we can still construct client-side preprocessing SPIR as follows:

- $\mathsf{Setup}(1^\lambda; 1^\lambda, D, R)$: the client privately chooses a random permutation $\pi : [n] \to [n]$ and the client and server use 2PC so that the client learns and outputs $\mathsf{st} = D \oplus (\pi \circ R)$ and the server learns nothing. Here, we use n-bit of randomness R and the notation $\pi \circ R = (R[\pi(1)], \ldots, R[\pi(n)])$.
- $\mathsf{Query}(\mathsf{st}, Q; D, R)$: on input $Q = (q_1, \ldots, q_k)$ consisting of k distinct indices, the client sends $\tau_u = (\pi(q_1), \ldots, \pi(q_k))$ and the server responds with $\tau_d = R[\tau_u]$. Then, the client outputs $\mathsf{ans} = \mathsf{st}[Q] \oplus \tau_d$.

The proofs of correctness, privacy, and data privacy are straightforward. Note that the indices in a batch must be distinct, as a repeated index would reveal the equality pattern of the batch to the server.

$O(k)$-bit client storage, $O(kn)$ communication. Next, we consider SPIR at the other end of the spectrum and show a SPIR with preprocessing which has optimal client storage but huge online communication.

- $\mathsf{Setup}(1^\lambda; 1^\lambda, D, R \in \{0,1\}^k)$: the client privately chooses a random PRF key $\mathsf{k} \in \{0,1\}^\lambda$ for a pseudorandom function $\mathsf{F}(\mathsf{k}, \cdot) : [k] \to \{0,1\}^n$ and the client and server use 2PC, so that the client learns the k-bit string C with $C[i] = \langle \mathsf{F}(\mathsf{k}, i), D \rangle \oplus R[i]$, where $\langle \cdot, \cdot \rangle$ denotes the inner product. Finally, the client outputs $\mathsf{st} = (\mathsf{k}, C)$.
- $\mathsf{Query}(\mathsf{st}, Q; D, R)$: on input $Q = (q_1, \ldots, q_k)$, the client parses $(\mathsf{k}, C) \leftarrow \mathsf{st}$ and uses each $e_{q_i} = 0^{q_i - 1} \| 1 \| 0^{n - q_i}$ to send $\tau_u = (\mathsf{F}(\mathsf{k}, i) \oplus e_{q_i})_{i \in [k]}$ to the server, and the server responds with $\tau_d = R \oplus (\langle \tau_u[i], D \rangle)_{i \in [k]}$. Then, the client outputs $\mathsf{ans} = C \oplus \tau_d$.

We omit the proof that the scheme is correct, private, and data private, but it is clear by construction that the sum of the inner products of vectors that are off by one will reveal the correct entry.

Existing 2-server SPIR. We additionally observe that we can convert any information-theoretic 2-server SPIR into a an offline-online SPIR without any public-key operations online, assuming the existence of generic 2PC. In particular, a client could obtain k independent responses from the first server under 2PC in the offline phase. Then, the client could make online second server queries (which do not require public-key operations, as they're information theoretically secure). This scheme would require $O(k \cdot C_1)$ state and $k \cdot C_2$ online communication, where C_i is the communication between the client and the i-th server for single query. However, this doesn't leverage the asymmetry between the offline and online phases — since most schemes to date have $C_1 = C_2$. Also, as far as the authors are aware, the best information-theoretic SPIR has $C_1 = C_2 = \tilde{O}(n^{1/3})$ [GIKM98].

If we instead consider computational 2-server SPIR, we could adapt approaches using 2-server computational PIR, e.g., those based on distributed point functions [GI14]. This approach is similar to our second toy construction, as we could one-time pad queries to the first server and then query the second server, who additionally adds the same pad to their responses. This will achieve better concrete efficiency than our toy construction above reducing the communication to be $k \cdot \text{polylog}(n)$.

However, all of these approaches require linear online server time. And, as it was done in [CK20] for PIR, we wish to show how to use offline preprocessing to achieve online *sublinear query time* for SPIR.

6.4 SPIR with Sublinear Online Computation

Our main construction adapts the recent works of [RMS24, HPPY25] to the SPIR setting. This construction achieves both online sublinear communication and computation per query — substantially improving over the efficiency of our simple constructions above. Additionally, it supports the maximum number of queries possible per offline phase (see Theorem 6.2) as well as it meets an optimal state/computation trade-off from prior work [Yeo23b], up to logarithmic factors. Our approach is not entirely generic, but it could possibly be applied to other PIR with client-side preprocessing; however, we critically use that the scheme does not require running parallel repetition to achieve a negligible failure rate.

For our high-level overview, we recall the way that recent PIRs with preprocessing schemes work at high level [CK20, CHK22, RMS24, ZPZS24, HPPY25, WR25]. We will then, give an overview of the difficulties and solutions that come up when augmenting these schemes with data privacy. Finally, we will give our more detailed construction.

PIR construction review. Simplifying a bit, the way that these schemes preprocess for their PIR queries is to select $h = \tilde{O}(\sqrt{n})$ random sets $S_1, \ldots, S_h$ each of size (about) $\sqrt{n}$. Then, the client will learn the XOR of each S_i for the database, i.e., $p_i \leftarrow \bigoplus_{x \in S_i} D[x]$, and finally the client then stores each (S_i, p_i).

In the online phase, to query index q, the client will find some set S_i with $q \in S_i$, which exists with high probability based on the choice of h. Then, the client sends $S' \leftarrow S_i \setminus q$ to the server, who returns $p' \leftarrow \bigoplus_{x \in S'} D[x]$ to the client. Finally, the client recovers $D[q] \leftarrow p' \oplus p_i$ correctly. Notice that this scheme requires large client storage; however, we can easily compress the representation of the random sets with a PRF, which we do below.

As written, the above has $\Theta(1/\sqrt{n})$ privacy for single query and no clear way to support multiple queries. But, all schemes to date operate in this overall paradigm of preprocessing by

finding the XOR of small random sets, choosing a set containing their query, and recovering the database element by asking the online server for the XOR of a modified set. The details of how they support multiple queries and achieve stronger privacy are scheme specific, and we will only illuminate these details later for our specific scheme.

Transforming PIR to SPIR. In order to transform this scheme from a PIR to a SPIR, we first need to modify the state so that it reveals no information about the database D. Since the sets $S_1, \ldots, S_h$ are already generated independently of the database, we just need to mask the parity bits $p_1, \ldots, p_h$. For all schemes like this, we can just add on a single one-time pad for each of these, so that they are actually computed as $p_i \leftarrow R[i] \oplus \bigoplus_{x \in S_i} D[x]$ for a uniform and independent $R[i]$. Additionally, symmetric PIR restricts *how* we can run the offline preprocessing, which is often either done by streaming or homomorphic encryption. So, we assume that the entire functionality of the streaming is run inside of a generic 2PC protocol, which provides privacy of the database (since the state is now independent of the database).

From this modification, it is straightforward to modify the online phase. All that is required is that the client, in addition to sending the set S', sends the index i for the hint that they used for the query. Then, the server can return $p' \leftarrow R[i] \oplus \bigoplus_{x \in S'} D[x]$, and the client can learn $D[q] \leftarrow p_i \oplus p'$ as before. This transformation will provide data privacy, because the client only downloaded one entry of information! So, even a malicious client will be unable to infer any more than a single database entry.

Dealing with complications. The three main details left out of the above transformation are: (1) how to compress set representations, (2) how to achieve negligible privacy, and (3) how to support multiple queries. In the next subsection, we give the details on how these are overcome with specific pseudocode. Here we provide the sketch of how we can lift the same ideas that were used in recent works [RMS24, HPPY25] from PIR to SPIR.

First, we modify the sets that are used for the hints. Instead of choosing random sets of size $\sqrt{n}$, we break up $[n]$ into $\sqrt{n}$ blocks of size $\sqrt{n}$ and restrict our sets so that that have a at most a single element from each block. We do this by choosing $\sqrt{n}/2 + 1$ random blocks and sampling a random offset (in $[\sqrt{n}]$) for each one. These elements constitute a hint set S and require us to store one PRF key for the partition sampling and one invertible PRF key per block to sample offsets, as was done in [HPPY25].[14] In the offline phase, the client will learn the (one-time padded) XOR of these sets just as before.

Now, to query the index q, the client again finds a hint set S_i which contains q. The client will remove q from S_i to get S' and sample fresh random offsets for the blocks which are not covered by S' (including the block that q belongs to). Then, the client will upload all of these offsets ordered by the blocks, together with the blocks covered by S' and the index i. Then, the server will send back the XOR of the values in these blocks as p'_1 in addition to the XOR of the values in the other blocks as p'_2, each padded with $R[i]$. Then, the first of which can be used with the precomputed parity to learn $D[q] \leftarrow p'_1 \oplus p_i$.

Unfortunately, as described, the above as two issues. First, it is not private if we reveal the blocks that S' covers to the server, since it allows the server to learn which blocks our query was certainly not in. Additionally, if we pad both p'_1 and p'_2 with the same value, then the data privacy is no longer preserved! To fix both of these problems, our scheme will with probability $1/2$ send

[14] As observed in prior works, it is only necessary for the client to choose a single master key for the entire state — from which they can derive other implicit keys.

the blocks that are not covered by S' rather than the blocks covered by S', which is effectively just permuting whether p'_1 or p'_2 is the relevant value to recover $D[q]$.

Additionally, we want to modify the online server so that it pads the p'_1 and p'_2 with different bits, so that the irrelevant p'_j is uniformly random and unknown to the client but that the relevant p'_j needs to remain hidden to the server for the client's privacy. To handle this tension, we can modify preprocessing so that, for each hint, the client privately selects whether it will send the real or fake blocks. Then, in the 2PC functionality, we have p_i set so that it is padded with either the left or right random bit depending on the choice (so that the other bit remains independent of the client state). Then online, the client will send send the real or fake blocks based on its offline choice rather than a new random choice. This way both the client's privacy and the server's data privacy are achieved.

Multiple queries. Fortunately, we can modify the techniques from prior work to achieve privacy even for adaptive queries, even though our definitions are written as batch queries. To do this, the key insight is to specify a way to *refresh hints*. At a high level, we cannot reuse hints while preserving privacy and removing hints from the pool of hints biases the distribution of queries in a noticeable way. To rectify this, in addition to preprocessing the hints, we compute a collection of "backup" hints.

Following prior work, these backup hints have a similar but different structure than the normal hints. Specifically, a backup hint consists of a set of size $\sqrt{n}/2$, based on a random partition of size $\sqrt{n}/2$ blocks and $\sqrt{n}/2$ offsets and the XOR of that set (padded with one of two random bits so that it gives no information about the database). After a client learns the value $D[q]$, we can refresh the hint that was used by finding a backup hint that does not cover the partition q. Then, we can take the compressed set representation (i.e., PRF key) S, and store $(S, q, p \oplus D[q])$, which represents the hint set $S \cup \{q\}$ and its padded parity as the refreshed hint. Then, this can be used and a normal hints and preserves the distribution of the preprocessed hints (since we removed one hint conditioned on it containing q and added back a hint conditioned on it containing q).

The only remaining issue with our scheme, is that we have to set the index of the hint that we send to the server. This is not an issue when making a single query, since it is distributed identically for any query q. However, for multiple queries, the hint index leaks information about the queries that have been made.[15] As a simple fix to this issue, the client can choose a secret random permutation π over their hints and backup hints. This way, every query just sends a distinct random index to the server no matter what the client queries.

This is essentially the entire sketch of our scheme, which can continue processing querying adaptively until it eventually runs out of backup hints. Next, we give a more detailed description of the above and give the code to support any specified k number of queries.

6.5 Detailed Construction

Here, we provide our detailed construction and theorem, with the appropriate details. We show how to handle k adaptive queries for any k, taking $h = k \cdot \text{polylog}(\lambda, n)$, so that with high probability the online phase can find valid hints for the online phase with all but negligible probability. This also involves modifying the sizes of the hint sets (to n/k rather than $\sqrt{n}$) for better efficiency.

[15] In particular, if a user queries for an index q which has already appeared in a previous query, then the index sent to the server is more likely to be larger (since it is more likely that the user must use a backup hint).

$\mathcal{F}(\mathsf{k}_p, \mathsf{k}_1, \ldots, \mathsf{k}_{n/k}, \pi, B; D, R)$

Parse $(r_{i,0}, r_{i,1})_{i\in[h+t]} \leftarrow R$

For each $i = 1, \ldots, h$: `// construct regular hints`

  $P \leftarrow \mathsf{F}_1(\mathsf{k}_p, i)$; $o_j \leftarrow \mathsf{F}_2(\mathsf{k}_j, i)$ for each $j \in [n/k]$

  $X \leftarrow \{(p-1)\cdot k + o_p : p \in P\}$

  $H[i] \leftarrow \bigoplus_{x\in X} D[x] \oplus r_{\pi(i),B[i]}$

For each $i = h+1, \ldots, h+t$: `// construct backup hints`

  $P \leftarrow \mathsf{F}_1(\mathsf{k}_p, i)$; $o_j \leftarrow \mathsf{F}_2(\mathsf{k}_j, i)$ for each $j \in [n/k]$

  $d \xleftarrow{\$} P$; $P \leftarrow P \setminus \{d\}$

  $X \leftarrow \{(p-1)\cdot k + o_p : p \in P\}$

  $H[i] \leftarrow (d, \bigoplus_{x\in X} D[x] \oplus r_{\pi(i),B[i]})$

Return $(H; \perp)$

Figure 1: Functionality for the setup phase of SPIR based on [RMS24, HPPY25].

We additionally assume that t (and therefore k) divides n without loss of generality, since one could always pad the database up to the nearest multiple of t.

- $\mathsf{Setup}(1^\lambda; 1^\lambda, D \in \{0,1\}^n, R \in \{0,1\}^{2(h+t)})$: the client privately samples keys $\mathsf{k}_p \leftarrow \{0,1\}^\lambda$ for PRF $\mathsf{F}_1(\mathsf{k}_p, \cdot) : [h+t] \to \binom{n/k}{n/2k+1}$, keys $\mathsf{k}_1, \ldots, \mathsf{k}_{n/k} \leftarrow \{0,1\}^\lambda$ for invertible PRF[16] $\mathsf{F}_2(\mathsf{k}, \cdot) : [h+t] \to [k]$, a secret permutation π over $[h+t]$, and $B \xleftarrow{\$} \{0,1\}^{h+t}$. Run secure MPC for functionality $\mathcal{F}$ specified in Figure 1, so the client learns H and then outputs $\mathsf{st} = (\mathsf{k}_p, \mathsf{k}_1, \ldots, \mathsf{k}_{n/k}, \pi, B, H)$.
- $\mathsf{Query}(\mathsf{st}, Q; D, R)$: on input $Q = (q_1, \ldots, q_k)$, the client parses $(\mathsf{k}_p, \mathsf{k}_1, \ldots, \mathsf{k}_{n/k}, \pi, B, H) \leftarrow \mathsf{st}$. Then, iteratively, the client runs the following query protocol for each $i \in [k]$:
  - The client picks a random unused (regular or refreshed) hint $H[i^*]$ which contains q_i (in block j), sets $(o_1, \ldots, o_{n/k}) \leftarrow (\mathsf{F}_2(\mathsf{k}_1, i^*), \ldots, \mathsf{F}_2(\mathsf{k}_{n/k}, i^*))$, partition set $P \leftarrow \mathsf{F}_1(\mathsf{k}_p, i^*) \setminus \{j\}$, and resamples $o_j \xleftarrow{\$} [k]$.

    The client sends $(o_1, \ldots, o_{n/k})$, $\pi(i^*)$, and P (if $B[i^*] = 0$) or its complement $\overline{P} = [n/k] \setminus P$ (if $B[i^*] = 1$).
  - The server receives offsets $o_1, \ldots, o_{n/k}$, index ι, and partition set P' and computes $X_0 \leftarrow \{(p-1)\cdot k + o_p : p \in P'\}$ and $X_1 \leftarrow \{(p-1)\cdot k + o_p : p \in [n/k] \setminus P'\}$.
    The server returns $s_b = \bigoplus_{x\in X_b} D[x] \oplus r_{\iota,b}$ for $b = 0, 1$.
  - The client computes $D[q_i] = H[i^*] \oplus s_{B[i^*]}$. For the next query, the client uses this result to refresh a backup hint. In particular, the client will find a backup hint $(d, x) \leftarrow H[\hat{i}]$ such that $j \notin \mathsf{F}_1(\mathsf{k}_p, \hat{i})$ or $j = d$. Then, the client stores $H[\hat{i}] \leftarrow (j, d, x \oplus D[q_i])$. For this hint, the client will construct the set P as $(\mathsf{F}_1(\mathsf{k}_p, \hat{i}) \setminus \{d\}) \cup \{j\}$.[17]
    At the end, the client outputs $(D[q_1], \ldots, D[q_k])$.

[16] See [HPPY25] for definitions and construction of invertible PRFs.

[17] We omit this in the above pseudocode for simplicity, see [HPPY25] for detailed pseudocode about managing backup hints.

We write the construction in the form of k rounds to illustrate that our construction can be used without batching. If instead one is specifically concerned about batching, one could figure out the offsets and partitions for refreshed backup hints before learning the results. Then, one can continue the loop before ever querying the server and send all k queries at once. Then, as long as the database entries are inferred in the correct order, the client can figure out all k values correctly.

Theorem 6.5 (SPIR construction)**.** *Assuming the existence of one-way functions and semi-honest generic 2PC, there exists a SPIR with preprocessing scheme for k queries on an n-bit database which:*

- *uses no online public-key operations,*
- *operates in a single round-trip,*
- *$\tilde{O}(k)$ state size,*
- *$\tilde{O}(n/k)$ per-query upload, $O(1)$ per-query download,*
- *and $\tilde{O}(n/k)$ per-query computation.*

Proof. The construction we analyze is described by the Setup and Query algorithms earlier in this section. Each of the claims about the scheme's efficiency can be verified by direct inspection of the scheme and the observation that the keys $\mathsf{k}_1, \ldots, \mathsf{k}_{n/k}$ can stored in $O(1)$ space and derived from a single master key in the client's memory. For claims about invertible PRF efficiency and security, see [HPPY25]. Throughout, we additionally assume that the 2PC protocol used is secure and correct with all but negligible probability.

For the purposes of this proof, we will call the hints which can be selected for a query "active" hints. For this first query, this is exactly the regular hints. When a hint is used, it is no longer considered active, and the backup hint that is refreshed is considered active. We will use the fact that for any of the k queries, the distribution of active hints is distributed identically with all but negligible probability. In particular, as long as the query function finds a suitable hint for the query and there is some backup hint that gets refreshed, the distribution is identical to that of the regular hints. And, both of these events occur with all but $\mathsf{negl}(\lambda, n)$ probability by our choice of $h = t = k \cdot \mathrm{polylog}(\lambda, n)$ and the pseudorandomness of the PRFs (see [RMS24]).

The protocol $\mathsf{Query}(\mathsf{st}, q; D, R)$ as described will be correct whenever some active hint contains the query q, which as mentioned happens with all but negligible probability.

The privacy of our protocol follows from the similar proofs given in prior works [RMS24, HPPY25], which we extend here. First, each offset sent to the server is distributed uniformly within the block and independently of the chosen index. This is because we only select a hint based on whether the specific offset for q is in the set. So, all other offsets are pseudorandom and independent if the PRF is secure, and the offset for q is replaced by a fresh random sample. The partition sent is private, because it is only selected based on if the relevant block is in the partition. But, then this block is removed and the partition is inverted with probability $1/2$. So, the server only sees a truly random partition of the n/k blocks. Although this inversion is chosen at preprocessing time by the client, the 2PC security ensures that the server cannot learn about which partition q belongs to. Finally, the indices sent to the server are k random values, drawn without replacement from $[h + t]$, because the permutation π is also kept private from the server and no hint is ever re-used.

More formally, consider an adversary $\mathcal{A}$ gets as input τ_{off} that is simulated from the server's inputs only, and assume the pseudorandom functions are replaced with truly random ones. Then, no matter what the input query set Q is, the uploaded transcript τ_{on} will appear as k messages each with n/k random offsets, a random partition of n/k blocks, and a uniformly random index

drawn from $[h+t]$ without replacement. Therefore, $\mathcal{A}$ cannot distinguish in this world between any two query sets Q_0 and Q_1.

Finally, data privacy is ensured by the security of the 2PC protocol. This is easy to see though. The download of the Query protocol is only 2 bits, and one of these bits is padded with a random value $r_{i,b}$ which is independent of the client's state. Because, in preprocessing, the client committed through their string B to one of the two $r_{i,0}, r_{i,1}$ values. The other value is not used to generate any part of the output H. Therefore, one of the two bits the client downloads is padded with a uniformly random and independent bit from the client's view, which therefore gives the client no information about the database.

A bit more formally, if we give an adversary $\mathcal{A}$, a state st and τ_{off} which is simulated using only the client's input and the output H of the 2PC protocol, then the adversary cannot distinguish between databases with $D[Q]$ fixed. Because, of their $2k$ bits downloaded in τ_{on}, k are independent of their view and the other k can be simulated from $D[Q]$ and H. And, therefore, the client can never distinguish between two databases D_0 and D_1 conditioned on their entries at Q being the same. □

For simplicity, we give definitions that are more in line with the syntax from our definition of PIR (Definition 3.7). However, our construction has some additional benefits that we do not make formal in this work.

Adaptive security. Our construction, although stated as a batch SPIR, is actually secure even if the client is allowed to issues its queries adaptively. This is because the client can refresh the hint it consumed at the end of each query, as is done in the adaptive schemes of [RMS24, HPPY25].

Online perfect data privacy and privacy. We comment that our scheme, as written, actually achieves perfect data privacy against adversaries without access to the offline transcript. This is essentially because the hints stored in st are all padded and give no information about the database until queried. Then, on each query, the client is only able to infer one of the database entries, as the other bit of information it receives is padded with a bit that is independent of st. Moreover, if we treat the pseudorandom functions as random oracles, then the queries are perfectly private against adversaries without the offline transcript. This shows that our scheme matches the limits shown in Lemma 6.3.

Acknowledgements

We'd like to thank David Cash for helpful suggestions pointing us toward prior results in big-key cryptography. We would also like to thank the anonymous reviewers for their helpful comments and suggestions, including one who pointed out a flaw in an earlier version of this work. We are grateful to Wei-Kai Lin, Shiyu Li, and Mingxun Zhou for pointing out a proof gap in our previous version regarding database encodings.

Part of this work was done while the second author was at Google, New York. The work of the second author at the Università di Salerno is partially supported by a Google Sponsored Research Agreement. Part of this work was done while the last author was at Columbia University that was partially supported by NSF grant CCF-2312242.

References

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45, Redwood Shores, CA, USA, August 13–15, 2003. Springer Berlin Heidelberg, Germany.

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy*, pages 962–979, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.

[ADN+10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 113–134, French Riviera, May 30 – June 3, 2010. Springer Berlin Heidelberg, Germany.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54, Santa Barbara, CA, USA, August 16–20, 2009. Springer Berlin Heidelberg, Germany.

[ADW10] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Survey: Leakage resilience and the bounded retrieval model. In Kaoru Kurosawa, editor, *ICITS 09: 4th International Conference on Information Theoretic Security*, volume 5973 of *Lecture Notes in Computer Science*, pages 1–18, Shizuoka, Japan, December 3–6, 2010. Springer Berlin Heidelberg, Germany.

[ALP+21] Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication-computation trade-offs in PIR. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 1811–1828. USENIX Association, August 11–13, 2021.

[ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 535–548, Berlin, Germany, November 4–8, 2013. ACM Press.

[ALSZ15] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 673–701, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany.

[App24] Apple. Getting up-to-date calling and blocking information for your app. https://developer.apple.com/documentation/sms_and_call_reporting/getting_up-to-date_calling_and_blocking_information_for_your_app, 2024.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th Annual ACM Symposium on Theory of Computing*, pages 479–488, Philadephia, PA, USA, May 22–24, 1996. ACM Press.

[BGI08] Eli Biham, Yaron J. Goren, and Yuval Ishai. Basing weak public-key cryptography on strong one-way functions. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 55–72, San Francisco, CA, USA, March 19–21, 2008. Springer Berlin Heidelberg, Germany.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1292–1303, Vienna, Austria, October 24–28, 2016. ACM Press.

[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, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland.

[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, Santa Barbara, CA, USA, August 20–24, 2000. Springer Berlin Heidelberg, Germany.

[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, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

[BKR16] Mihir Bellare, Daniel Kane, and Phillip Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 373–402, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

[BM09] Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 374–390, Santa Barbara, CA, USA, August 16–20, 2009. Springer Berlin Heidelberg, Germany.

[BMW24] Alexander Burton, Samir Jordan Menon, and David J. Wu. Respire: High-rate PIR for databases with small records. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024: 31st Conference on Computer and Communications Security*, pages 1463–1477, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.

[BPSY24] Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. Batch PIR and labeled PSI with oblivious ciphertext compression. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024: 33rd USENIX Security Symposium*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association.

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[CG97] Benny Chor and Niv Gilboa. Computationally private information retrieval (extended abstract). In *29th Annual ACM Symposium on Theory of Computing*, pages 304–313, El Paso, TX, USA, May 4–6, 1997. ACM Press.

[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, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press.

[CHK22] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-server private information retrieval with sublinear amortized time. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in*

Computer Science, pages 3–33, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[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, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

[CIMR25] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. Secret-key PIR from random linear codes. Cryptology ePrint Archive, Report 2025/646, 2025.

[CK20] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 44–75, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 68–86, Warsaw, Poland, May 4–8, 2003. Springer Berlin Heidelberg, Germany.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414, Prague, Czech Republic, May 2–6, 1999. Springer Berlin Heidelberg, Germany.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

[DG15] Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 577–584, Portland, OR, USA, June 14–17, 2015. ACM Press.

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[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, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 65–87, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer Berlin Heidelberg, Germany.

[DKL+23] Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 417–446, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[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, Bruges, Belgium, May 14–18, 2000. Springer Berlin Heidelberg, Germany.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science*, pages 293–302, Philadelphia, PA, USA, October 25–28, 2008. IEEE Computer Society Press.

[DRRT18] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. PIR-PSI: Scaling private contact discovery. *Proceedings on Privacy Enhancing Technologies*, 2018(4):159–178, October 2018.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer Berlin Heidelberg, Germany.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324, Cambridge, MA, USA, February 10–12, 2005. Springer Berlin Heidelberg, Germany.

[FLLP24] Ben Fisch, Arthur Lazzaretti, Zeyu Liu, and Charalampos Papamanthou. ThorPIR: Single server PIR via homomorphic thorp shuffles. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024: 31st Conference on Computer and Communications Security*, pages 1448–1462, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. 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 438–464, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 640–658, Copenhagen, Denmark, May 11–15, 2014. Springer Berlin Heidelberg, Germany.

[GIKM98] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *30th Annual ACM Symposium on Theory of Computing*, pages 151–160, Dallas, TX, USA, May 23–26, 1998. ACM Press.

[GLM16] Matthew Green, Watson Ladd, and Ian Miers. A protocol for privately reporting ad impressions at scale. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1591–1601, Vienna, Austria, October 24–28, 2016. ACM Press.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815, Lisbon, Portugal, July 11–15, 2005. Springer Berlin Heidelberg, Germany.

[GZS24] Ashrujit Ghoshal, Mingxun Zhou, and Elaine Shi. Efficient pre-processing PIR without public-key cryptography. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 210–240, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[GZSP25] Ashrujit Ghoshal, Mingxun Zhou, Elaine Shi, and Bo Peng. Pseudorandom functions with weak programming privacy and applications to private information retrieval. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part VII*, volume 15607 of *Lecture Notes in Computer Science*, pages 284–313, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[HDCGZ23] Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. Private web search with tiptoe. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 396–416, 2023.

[HOWW19] Ariel Hamlin, Rafail Ostrovsky, Mor Weiss, and Daniel Wichs. Private anonymous data access. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 244–273, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.

[HPPY25] Alexander Hoover, Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Plinko: Single-server PIR with efficient updates via invertible PRFs. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part VI*, volume 15606 of *Lecture Notes in Computer Science*, pages 3–33, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[HV16] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 397–429, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer Berlin Heidelberg, Germany.

[IKOS04] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch codes and their applications. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 262–271, Chicago, IL, USA, June 13–16, 2004. ACM Press.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[ISW24] Yuval Ishai, Elaine Shi, and Daniel Wichs. PIR with client-side preprocessing: Information-theoretic constructions and lower bounds. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part IX*, volume 14928 of *Lecture Notes in Computer Science*, pages 148–182, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[KC21] Dmitry Kogan and Henry Corrigan-Gibbs. Private blocklist lookups with checklist. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 875–892. USENIX Association, August 11–13, 2021.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 54–70, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall, CRC Press, third edition, 2014.

[KNR95] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. In *27th Annual ACM Symposium on Theory of Computing*, pages 596–605, Las Vegas, NV, USA, May 29 – June 1, 1995. ACM Press.

[KNR99] I. Kremer, N. Nisan, and D. Ron. On randomized one-round communication complexity. *Computational Complexity*, 8:21–49, June 1999.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[KRS+19] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile private contact discovery at scale. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019: 28th USENIX Security Symposium*, pages 1447–1464, Santa Clara, CA, USA, August 14–16, 2019. USENIX Association.

[LKLM21] Kristin Lauter, Sreekanth Kannepalli, Kim Laine, and Radames Cruz Moreno. Password Monitor: Safeguarding passwords in Microsoft Edge. https://www.microsoft.com/en-us/research/blog/password-monitor-safeguarding-passwords-in-microsoft-edge/, 2021.

[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, Orlando, FL, USA, June 20–23, 2023. ACM Press.

[LMW25] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Black box crypto is useless for doubly efficient PIR. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part VI*, volume 15606 of *Lecture Notes in Computer Science*, pages 65–93, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78, Barcelona, Spain, May 20–24, 2007. Springer Berlin Heidelberg, Germany.

[LP23a] Arthur Lazzaretti and Charalampos Papamanthou. Near-optimal private information retrieval with preprocessing. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023: 21st Theory of Cryptography Conference, Part II*, volume 14370 of *Lecture Notes in Computer Science*, pages 406–435, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland.

[LP23b] Arthur Lazzaretti and Charalampos Papamanthou. TreePIR: Sublinear-time and polylog-bandwidth private information retrieval from DDH. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 284–314, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[LP24] Arthur Lazzaretti and Charalampos Papamanthou. Single pass client-preprocessing private information retrieval. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024: 33rd USENIX Security Symposium*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association.

[LPA+19] Lucy Li, Bijeeta Pal, Junade Ali, Nick Sullivan, Rahul Chatterjee, and Thomas Ristenpart. Protocols for checking compromised credentials. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 1387–1403, London, UK, November 11–15, 2019. ACM Press.

[LV16] Tianren Liu and Vinod Vaikuntanathan. On basing private information retrieval on NP-hardness. 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 372–386, Tel Aviv, Israel, January 10–13, 2016. Springer Berlin Heidelberg, Germany.

[LWZW25] Shuaishuai Li, Anyu Wang, Cong Zhang, and Xiaoyun Wang. Two-server sublinear PIR with symmetric privacy and statistical security. Cryptology ePrint Archive, Report 2025/1313, 2025.

[MCR21] Muhammad Haris Mughees, Hao Chen, and Ling Ren. OnionPIR: Response efficient single-server PIR. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 2292–2306, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.

[Mer78] Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[MR23] Muhammad Haris Mughees and Ling Ren. Vectorized batch private information retrieval. In *2023 IEEE Symposium on Security and Privacy*, pages 437–452, San Francisco, CA, USA, May 21–25, 2023. IEEE Computer Society Press.

[MVM24] Miti Mazmudar, Shannon Veitch, and Rasoul Akhavan Mahdavi. Peer2pir: Private queries for ipfs. *arXiv preprint arXiv:2405.17307*, 2024.

[MW22] Samir Jordan Menon and David J. Wu. SPIRAL: Fast, high-rate single-server PIR via FHE composition. In *2022 IEEE Symposium on Security and Privacy*, pages 930–947, San Francisco, CA, USA, May 22–26, 2022. IEEE Computer Society Press.

[MZRA22] Yiping Ma, Ke Zhong, Tal Rabin, and Sebastian Angel. Incremental offline/online PIR. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 1741–1758, Boston, MA, USA, August 10–12, 2022. USENIX Association.

[OS07] Rafail Ostrovsky and William E. Skeith, III. A survey of single-database private information retrieval: Techniques and applications (invited talk). In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 393–411, Beijing, China, April 16–20, 2007. Springer Berlin Heidelberg, Germany.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer Berlin Heidelberg, Germany.

[PPY18] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Private stateful information retrieval. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1002–1019, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

[PSY23] Sarvar Patel, Joon Young Seo, and Kevin Yeo. Don't be dense: Efficient keyword PIR for sparse databases. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023: 32nd USENIX Security Symposium*, pages 3853–3870, Anaheim, CA, USA, August 9–11, 2023. USENIX Association.

[PT20] Jeongeun Park and Mehdi Tibouchi. SHECS-PIR: Somewhat homomorphic encryption-based compact and scalable private information retrieval. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020: 25th European Symposium on Research in Computer Security, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 86–106, Guildford, UK, September 14–18, 2020. Springer, Cham, Switzerland.

[PY22] Giuseppe Persiano and Kevin Yeo. Limits of preprocessing for single-server PIR. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *33rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2522–2548, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022. ACM-SIAM.

[RMS24] Ling Ren, Muhammad Haris Mughees, and I Sun. Simple and practical amortized sublinear private information retrieval using dummy subsets. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024: 31st Conference on Computer and Communications Security*, pages 1420–1433, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.

[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, Cambridge, MA, USA, February 19–21, 2004. Springer Berlin Heidelberg, Germany.

[SACM21] Elaine Shi, Waqar Aqeel, Balakrishnan Chandrasekaran, and Bruce M. Maggs. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 641–669, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[SKI03] SeongHan Shin, Kazukuni Kobara, and Hideki Imai. Leakage-resilient authenticated key establishment protocols. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 155–172, Taipei, Taiwan, November 30 – December 4, 2003. Springer Berlin Heidelberg, Germany.

[SWZ24] Jaspal Singh, Yu Wei, and Vassilis Zikas. Information-theoretic multi-server private information retrieval with client preprocessing. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part IV*, volume 15367 of *Lecture Notes in Computer Science*, pages 423–450, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[TPY+19] Kurt Thomas, Jennifer Pullman, Kevin Yeo, Ananth Raghunathan, Patrick Gage Kelley, Luca Invernizzi, Borbala Benko, Tadek Pietraszek, Sarvar Patel, Dan Boneh, and Elie Bursztein. Protecting accounts from credential stuffing with password breach alerting. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019: 28th USENIX Security Symposium*, pages 1556–1571, Santa Clara, CA, USA, August 14–16, 2019. USENIX Association.

[WR25] Zhikun Wang and Ling Ren. Single-server client preprocessing PIR with tight space-time trade-off. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part VI*, volume 15606 of *Lecture Notes in Computer Science*, pages 94–122, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

[Yeo23a] Kevin Yeo. Cuckoo hashing in cryptography: Optimal parameters, robustness and applications. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 197–230, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[Yeo23b] Kevin Yeo. Lower bounds for (batch) PIR with private preprocessing. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 518–550, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[YP21] Kevin Yeo and Sarvar Patel. Protecting your device information with private set membership, 2021.

[ZLTS23] Mingxun Zhou, Wei-Kai Lin, Yiannis Tselekounis, and Elaine Shi. Optimal single-server private information retrieval. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology*

– *EUROCRYPT 2023, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 395–425, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[ZPZS24] Mingxun Zhou, Andrew Park, Wenting Zheng, and Elaine Shi. Piano: Extremely simple, single-server PIR with sublinear server computation. In *2024 IEEE Symposium on Security and Privacy*, pages 4296–4314, San Francisco, CA, USA, May 19–23, 2024. IEEE Computer Society Press.

A Communication Costs for Few Queries

We note that many of our lower bounds show that PIR with preprocessing requires $t = \Omega(n/s)$ amortized communication (see Section 5). However, all of these only apply to constructions in our model (in particular, with oracle-free encodings) with s-bit client storage that support at least $k = \Omega(s)$ queries per offline phase. One may wonder, what is the communication complexity of schemes that potentially support a small and bounded number of queries per offline phase where our lower bound no longer applies? It turns out that our lower bound is not applicable for good reason as there exist constructions with smaller communication in this case.

In particular, the recent works of Ghoshal et al. [GZS24, GZSP25] show that this communication limit is not inherent. They provide PIR which achieve $\tilde{O}(\sqrt{n/s})$ and $\mathrm{polylog}(\lambda, n/s)$ (respectively) per-query communication without re-initializing the state. Re-initializing the state costs amortized $O(n/s)$ download, matching our lower bound. We leave it as an open problem to prove a lower bound on the upload and download trade-offs for constructions that support only $k \leq s$ queries.

B Generalization to Resampleable Oracles

A very helpful anonymous reviewer pointed out that our techniques are not limited to crypto oracles. Prior works looking at crypto oracles "compile out" their oracle at some point in their proof [DH24, LMW25]. This leads to a setting where an honest party can simulate the oracle for the entire protocol.

In contrast, our results require no such transformation. We only require that the adversary can sample oracles O' and check for those which are *consistent* with a given set of q pairs of query inputs and outputs $S = \{(\mathsf{in}_1, \mathsf{out}_1), \ldots, (\mathsf{in}_q, \mathsf{out}_q)\}$. In fact, our proof does not even require that the adversary can do this efficiently, only that it can be done in some bounded amount of time. From the main body of the paper, Lemma 3.6 shows that crypto oracles from prior works fall into this class.

More precisely, all we need is that the (random) oracle O satisfies the following definition of resampleability.

Definition B.1 (Resampleable Oracles)**.** *We say an oracle (random variable)* O *is* (q, δ)-resampleable *if there exists a (possibly inefficient) algorithm* $\mathcal{A}$ *such that for all sets* $S = \{(\mathsf{in}_i, \mathsf{out}_i)\}_{i \in [q]}$*, we have that* $\mathsf{O}' \leftarrow \mathcal{A}(S)$ *and*

$$\Delta(\mathsf{O}', \mathsf{O} \mid \forall_{i \in [q]}\ \mathsf{O}(\mathsf{in}_i) = \mathsf{out}_i) < \delta.$$

In other words, $\mathcal{A}$ *can resample an oracle that is* δ*-close (in statistical distance) to a fresh sample of* O *conditioned on the event that* O *is consistent with the set of queries* S*.*

From this definition, it is clear that the construction in Section 4.2 and Theorem 4.1 apply to this class of oracles and not only to crypto oracles, under the same restriction that the encoding algorithm Init makes no oracle queries (Definition 3.8). This is just because we can appeal directly to the definition everywhere that we otherwise would've appealed to Lemma 3.6, as long as $q = \text{poly}(\lambda)$ and $\delta \leq 2^{-(4n+2s+1)}$. Here, the bound on δ comes from the fact that Hint and Recon resample the oracle at most $2^{3n+2s+1}$ times in total and a hybrid argument charges δ per resampling, so the total statistical distance from the crypto oracle case is at most 2^{-n}, which only changes the 2^{-n+1} term of Theorem 4.3 by a constant factor. Note that δ must be exponentially small in n; negligible in λ does not suffice.

Remark B.2. *While this is a generalization of our results, at the time of writing, the authors are unaware of any interesting oracle class* O *that is not a crypto oracle but is still useful for cryptography. Depending on which oracles are shown to be* (q, δ)*-resampleable, this definition may be useful for showing further impossibility results or general barriers in constructing cryptography.*