PIR with Client-Side Preprocessing: Information-Theoretic Constructions and Lower Bounds

Yuval Ishai
Technion

Elaine Shi
CMU

Daniel Wichs
Northeastern University
and NTT Research

January 1, 2025

Abstract

It is well-known that classical Private Information Retrieval (PIR) schemes without preprocessing must suffer from linear server computation per query. Moreover, any such single-server PIR with sublinear bandwidth must rely on public-key cryptography. Several recent works showed that these barriers pertaining to classical PIR can be overcome by introducing a preprocessing phase where each client downloads a small hint that helps it make queries subsequently. Notably, the Piano PIR scheme (and subsequent improvements) showed that with such a client-side preprocessing, not only can we have PIR with sublinear computation and bandwidth per query, but somewhat surprisingly, we can also get it using only symmetric-key cryptography (i.e., one-way functions).

In this paper, we take the question of minimizing cryptographic assumptions to an extreme. In particular, we are the first to explore the landscape of *information theoretic* single-server preprocessing PIR. We make contributions on both the upper- and lower-bounds fronts. First, we show new information-theoretic constructions with various non-trivial performance tradeoffs between server computation, client space and bandwidth. Second, we prove a (nearly) tight lower bound on the tradeoff between the client space and bandwidth in information-theoretic constructions. Finally, we prove that any computational scheme that overcomes the information-theoretic lower bound and satisfies a natural syntactic requirement (which is met by all known constructions) would imply a hard problem in the complexity class SZK. In particular, this shows that Piano achieves (nearly) optimal client space and bandwidth tradeoff subject to making a black-box use of a one-way function. Some of the techniques we use for the above results can be of independent interest.

Contents

1 Introduction

Private Information Retrieval (PIR), originally proposed by Chor, Goldreich, Kushilevitz, and Sudan [CGKS95], allows a client to retrieve a record from a database hosted by one or more untrusted servers, without revealing which record was fetched. PIR promises a rich spectrum of applications, such as private web search [HDCG$^+$23], private contact discovery [sig], private DNS [pri], private detection of whether a user's password is contained in a leaked password database [hav], private lightweight clients for cryptocurrencies, and so on. Classical PIR schemes [CGKS95, CG97, CMS99, GR05, KO97, Lip09, OS07, Gas04, BFG03, SC07, OG11, MCG$^+$08, MG07, HHCG$^+$23, MW22] do not use any preprocessing. Unfortunately, in the classical model, it is known that any query must inherently take linear server computation [BIM00]: intuitively, if there is some record that the server did not look at, it leaks the fact that the client is not looking for that particular record. This linear computation lower bound is viewed as a significant barrier towards scaling PIR to real-world data sizes (e.g., private DNS and private web search). Furthermore, it is known that single-server PIR schemes with sublinear bandwidth in the classical model cannot exist information-theoretically and in fact require strong "cryptomania" assumptions [DMO00].

The paradigm shift to preprocessing PIR. To overcome the linear computation barrier, a recent line of works, pioneered by Beimel et al. [BIM00] and Corrigan-Gibbs and Kogan [CK20], proposed new preprocessing models for PIR schemes. In the most general form, the client and the server can engage in some preprocessing protocol before any queries arrive, and at the end of the preprocessing protocol, the client and the server can each store some extra state which later facilitates efficient queries. Unless otherwise noted, we require that *after a single preprocessing upfront, the PIR scheme should support an arbitrary number of queries* (also called the *unbounded* query model). Beimel et al. [BIM00] and Corrigan-Gibbs and Kogan [CK20] considered a couple special forms of preprocessing. Specifically, Beimel et al. [BIM00] considered a *public preprocessing* that did not involve the client, where the server encodes its database into some redundant form. The advantage of this approach is the same preprocessing works for all clients. By contrast, Corrigan-Gibbs and Kogan [CK20] considered a *client-specific preprocessing* model where every client engages in a preprocessing protocol with the server, at the end of which the client obtains a hint that will facilitate efficient queries subsequently. The server, on the other hand, stores only the original database and need not store per-client state.

Several recent works [BIM00, WY05, PPY18, CHR17, BIPW17, CK20, SACM21, CHK22, ZLTS23, LMW23, ZPSZ24, MIR23, GZS24a] study various models of PIR with a preprocessing phase, and show that they can overcome the linear computation barrier for classical PIR. Not only so, very recent works [ZPSZ24, MIR23, GZS24a] also showed that PIR with client-specific preprocessing is possible with weaker cryptographic assumptions than classical PIR. Specifically, relying only on one-way functions (OWFs) and without any use of public-key cryptography, we can construct single-server preprocessing PIR schemes with $O(\sqrt{n})$ bandwidth, $\widetilde{O}(\sqrt{n})$ computation, and requiring only $\widetilde{O}(\sqrt{n})$ client space [ZPSZ24, MIR23, GZS24a]. This may initially seem surprising because in classical land, it is known that any single-server PIR scheme with sublinear bandwidth would imply oblivious transfer [DMO00] (even when we allow unbounded polynomial computation and space). The reason why we can get away with weaker assumptions in the preprocessing model is because known schemes [ZPSZ24, MIR23, GZS24a] as well as the constructions in our paper allow a preprocessing phase with linear communication which can be amortized over an unbounded number of subsequent queries. Recent works [ZPSZ24, MIR23, GZS24a] showed that minimizing cryptographic assumptions also comes with the benefit of yielding concretely efficient constructions which, in many realistic settings, outperform the state-of-the-art classical PIR schemes [MW22, HHCG+23, DPC23, LMRS23] by at least two orders of magnitude in terms of response time.

1.1 Our Results and Contributions

In this paper, we take the question of minimizing cryptographic assumptions to an extreme, studying the possibility of *information-theoretic* security for preprocessing PIR with non-trivial performance bounds. We also explore the necessary cryptographic assumptions in cases where computational security is the best one can hope for.

In the two-server setting, the question of preprocessing PIR with information-theoretic security was previously studied by Beimel et al. [BIM00] and by Woodruff and Yekhanin [WY05], who showed non-trivial information-theoretic preprocessing PIR schemes with sublinear computation (see Table 1). One interesting question is whether we can get non-trivial information-theoretic preprocessing PIR in the single-server setting. Unfortunately, the overall algorithmic framework from [BIM00, WY05] fundamentally cannot work for the single-server setting, since their schemes adopt the public preprocessing model where the clients do not store any state resulting from the preprocessing. In this model, the lower bound of Di Crescenzo et al. [DMO00] still holds, and it is impossible to get any information-theoretic single-server PIR with sublinear bandwidth, despite the server-side preprocessing.

To the best of our knowledge, we are the first to explore information-theoretic preprocessing PIR in the single-server setting. Our results include both new upper bounds and lower bounds. To circumvent the aforementioned barrier for public preprocessing, we focus on the client-specific preprocessing model — in this sense, our algorithmic techniques depart significantly from those of [BIM00, WY05]. We also prove a (nearly) tight lower bound that characterizes the client space and bandwidth tradeoff in the information-theoretic setting, and gives a barrier for improving this tradeoff using only "symmetric" cryptographic assumptions. We summarize our main results below.

Lower bound on client space and bandwidth tradeoff. We prove the following lower bound that characterizes the client space and bandwidth tradeoff for preprocessing PIR schemes.

Theorem 1.1 (Client space and bandwidth tradeoff for 1-server preprocessing PIR)**.** *For any $t \geq 0$, there is no information-theoretic preprocessing PIR scheme with less than $t/2$ client space, where a consecutive sequence of t queries altogether consume less than $n/2$ bandwidth. Moreover, if there is such a computationally secure scheme then one-way functions (OWFs) must exist.*

In particular, the conclusions of Theorem 1.1 apply to any scheme that supports at least $t = \sqrt{n}$ online queries, where the client space and average per-query bandwidth are $o(\sqrt{n})$.

The above lower bound holds in a very strong sense. It holds even when we allow: (1) general preprocessing that arbitrarily encodes the data on the server and generates a hint of the client, without imposing any limits on the server space or the computation/communication of the preprocessing, (2) both the server and the client can maintain arbitrary dynamic state after the preprocessing; the lower-bound is only on the size of the initial client hint, (3) the scheme can use unbounded polynomial server/client computation per query and arbitrary number of roundtrips. Our lower bound also implies a fundamental client space and bandwidth tradeoff for any *truly* information-theoretic Oblivious RAM (ORAM) scheme — here "truly information-theoretic" means that encryption is no longer considered a free assumption, to disambiguate from the standard "statistically secure" ORAM literature [Ajt10, DMN11, SCSL11, SvDS+13] where encryption is considered as a free assumption.[1]

[1]The main difference between preprocessing PIR and ORAM is that ORAM allows the server to store a linear

Table 1: Information-theoretic preprocessing PIR: our results and comparisons. The server space counts only the extra storage besides storing the original database. The *online* bandwidth is on the critical path for the client to obtain the answer to its query, whereas the *offline* bandwidth can be viewed as background maintenance work. The lower bounds hold for any $t \geq 1$. See also Section 1.3 for additional related work.

Scheme	Compute	Comm.	Space		#
			client	server	servers
Prior work					
[BIM00]	$O(n/\log^2 n)$	$O(n^{1/3})$	0	$O(n^2)$	2
[WY05]	$O(n/\mathsf{poly}\log n)$	$O(n^{1/3})$	0	$\mathsf{poly}(n)$	2
[BIM00]	$O(n^{1/2+\epsilon})$	$O(n^{1/2+\epsilon})$	0	$O(n^{1+\epsilon'})$	2
Lower bound ([CK20])	$\Omega(n/t)$	any	$O(t)$	0	1
Our results					
Lower bound (Section 5)	any	$\Omega(n/t)$	$O(t)$	any	1
Theorem 3.1	$n^{1+o(1)}$	$n^{o(1)}$ online $O(n^{1/2})$ offline	$n^{1/2+o(1)}$	0	1
Theorem A.3	$\widetilde{O}(n^{1/2})$ client $\widetilde{O}(n^{2/3})$ server	$O(n^{1/3})$ online $O(n^{1/2})$ offline	$\widetilde{O}(n^{2/3})$	0	1
Theorem B.1	$\widetilde{O}(n^{1/2})$ client $\widetilde{O}(n^{2/3})$ server	$O(n^{1/3})$	$\widetilde{O}(n^{2/3})$	0	2

The lower bound is also *tight* up to an $n^{o(1)}$ factor, since if we do not care about the computation overhead being sublinear, we can indeed construct an information-theoretic PIR scheme with $t \cdot n^{o(1)}$ client space and $O(n/t)$ amortized bandwidth per query (see Theorem 1.3 and Section 3.1).

We now compare our lower bound result to prior lower bounds. Previous works have shown lower bounds that characterize client space and server computation tradeoff [CK20, CHK22] (in even computational schemes). Further, the elegant work of Yeo extended these lower bounds to the batched setting [Yeo23]. To the best of our knowledge, we are the first to prove lower bounds that characterize the tradeoff between client space and bandwidth for preprocessing PIR. Note that since the bandwidth is a lower bound on the server computation, in fact, our lower bound also characterizes the tradeoff between server computation and client space. However, the nature of the new lower bound is incomparable to prior ones [CK20, CHK22, Yeo23]. Specifically, prior lower bounds show a tradeoff between server computation and client space which applies even for computational schemes, but restricted to the setting where the server stores only the original database and nothing else. Our lower bound shows that any scheme with an improved tradeoff implies OWF and this holds even when the server can store a preprocessed version of the database.

SZK barrier for database-oblivious schemes. We get an even stronger limitation for *database-oblivious* preprocessing PIR schemes, where the client's questions to the server depend only on its randomness and the index sequence and not on the database-dependent answers. Such schemes

amount of dynamic per-client state, whereas most known PIR schemes want the server storage to be independent of the number of clients and static.

can be implemented with a single roundtrip. All of the PIR schemes constructed in this paper, as well as all prior related works [CK20, SACM21, ZPSZ24, MIR23, LP22] are database oblivious. For such database-oblivious schemes, we get a variant of Theorem 1.1 with the stronger conclusion that there is an average-case hard promise problem in the complexity class SZK (the class of problems with statistical zero-knowledge proofs).[2] Further, this lower bound shows that the client-space and bandwidth tradeoff of Piano [ZPSZ24] and subsequent improvements [MIR23, GZS24a] is optimal up to a super-logarithmic factor for schemes in Minicrypt. Note that although Ghoshal et al. [GZS24a] improve the online bandwidth to $\widetilde{O}(n^{1/4})$ under $\widetilde{O}(n^{1/2})$ client space, their offline bandwidth is still $\widetilde{O}(n^{1/2})$. Our SZK barrier also shows the impossibility of improving the *offline* bandwidth of Goshal et al. [GZS24a] if we restrict ourselves to black-box use of OWF or even stronger symmetric primitives.

Theorem 1.2 (SZK barrier for preprocessing PIR)**.** *Any (1-roundtrip) database-oblivious preprocessing PIR scheme with less than $t/3$ client space, and a consecutive sequence of t queries which together consume less than $n/3$ bandwidth, implies an average-case hard promise problem in SZK. In particular, it implies a separation of (promise) SZK from BPP.*

A hard SZK problem cannot be constructed in a black-box way from a OWF or even a collision-resistant hash function [KY18, BD19, BDV21], and implies strong cryptographic primitives such as constant-round dual-mode commitments [OV08]. More importantly, all known concrete candidates for such promise problems are based on "structured" assumptions [BDRV18], and no random oracle construction is known. Consequently, any database-oblivious PIR scheme which circumvents the lower bounds of Theorem 1.1 is effectively closer to "cryptomania" than to "minicrypt." Finally, since our proof of Theorem 1.2 fully relativizes, we also get a barrier for preprocessing PIR in the *random oracle model*. Concretely, an unconditional ROM construction with $o(\sqrt{n})$ client space and bandwidth would separate (promise) SZK from BPP relative to a random oracle, settling a natural open problem in complexity theory. We are not aware of any previous application of such an SZK barrier to give evidence against constructions of a natural cryptographic primitive from any "symmetric" primitive.

Constructions of information-theoretic preprocessing PIR. Our first construction achieves a nearly optimal tradeoffs between client space and overall per-query bandwidth, allowing both to be $O(n^{1/2+o(1)})$. However, this comes at the cost of requiring slightly super-linear server computation per query. This result shows the approximate tightness of the lower bound in Theorem 1.1. We state the results below.

Theorem 1.3 (1-server PIR with nearly optimal client space and bandwidth tradeoff)**.** *For every $t \geq 1$, there exists a single-server preprocessing PIR scheme that supports an* **unbounded** *number of queries with the following performance bounds:*

- *there is an initial preprocessing in which the client makes a streaming pass over the database consuming $O(n)$ bandwidth and server computation, and moreover, the client performs $n^{1+o(1)} \cdot t$ computation;*
- *after the preprocessing, each query requires $n^{o(1)}$ online bandwidth, $O(n/t)$ offline bandwidth, and $n^{1+o(1)}$ client and server computation;*
- *the client space is $t \cdot n^{o(1)}$;*

[2]In fact, our conclusion is even stronger: we get a variant of a somewhere statistical binding (SSB) hash function [HW15], which implies a hard problem in SZK but is not known to be implied by it.

In the above, online bandwidth captures the communication on the critical path for the client to obtain an answer, and offline communication can be viewed as background maintenance work.

To get Theorem 1.3, we adapt the Dvir-Gopi 2-server PIR scheme [DG16] to a 1-server preprocessing PIR scheme. The details are described in Section 3.1.

Our second upper bound result gives a 1-server preprocessing PIR scheme with sublinear server computation $\widetilde{O}(n^{2/3})$. However, this comes at the cost of having a slightly worse trafeoff between client space and overall per-query bandwidth: we can get bandwitdth $\widetilde{O}(n^{1/2})$, but the client space is now $\widetilde{O}(n^{2/3})$. We state the result in the following theorem.

Theorem 1.4 (1-server PIR with sublinear computation)**.** *There exists an information theoretic 1-server preprocessing PIR scheme supporting unbounded queries, and achieving the following performance bounds with all but* $\mathsf{negl}(n)$ *probability where* $\widetilde{O}(\cdot)$ *hides a multiplicative* $\log n \cdot \alpha(n)$ *factor for an arbitrary super-constant function* $\alpha(n)$*:*

- *there is an initial preprocessing phase in which the client makes a streaming pass over the database consuming* $O(n)$ *bandwidth and server computation; and moreover, the client performs* $\widetilde{O}(n)$ *computation;*
- *after the initial preprocessing, every query takes* $\widetilde{O}(n^{1/3})$ online *bandwidth,* $\widetilde{O}(n^{1/2})$ offline *bandwidth,* $\widetilde{O}(n^{1/2})$ *client computation, and* $\widetilde{O}(n^{2/3})$ *server computation; and*
- *the client space is* $\widetilde{O}(n^{2/3})$*.*

In the sublinear-server-computation regime, there is still some gap between our upper bound (Theorem 1.4) and the lower bounds (Theorem 1.1, [CK20, Theorem 23]). It is a fascinating open problem whether it is possible to close this gap and achieve a scheme where the client space, bandwidth and server computation are all bounded by $\widetilde{O}(n^{1/2})$.

For all of our upper bound results, we can also think of the results as not having any "preprocessing phase", but instead count the cost of preprocessing in the cost of the first query. This results in schemes where the worst-case bandwidth/communication is large, but we can amortize this cost over arbitrarily many subsequent queries. Therefore we get a scheme without preprocessing that achieves the above bandwidth/computation efficiency for the queries in an amortized sense, while maintaining the same bounds on client space in the worst-case.

As a by-product of our work, we also improve the theoretical state-of-the-art for 2-server preprocessing PIR in the information-theoretic setting. Specifically, we can adapt the construction of Theorem 1.4 into a 2-server preprocessing PIR as stated in the following corollary:

Theorem 1.5 (2-server PIR with sublinear computation)**.** *There exists an information theoretic 2-server PIR scheme that supports unbounded queries, and achieving the following performance bounds:*

- *there is an initial preprocessing that requires* $\widetilde{O}(n^{2/3})$ *bandwidth,* $\widetilde{O}(n^{2/3})$ *client computation, and* $\widetilde{O}(n)$ *server computation;*
- *after the initial preprocessing, every query takes* $\widetilde{O}(n^{1/3})$ *online bandwidth,* $\widetilde{O}(n^{1/6})$ *offline bandwidth,* $\widetilde{O}(n^{1/2})$ *client computation, and* $\widetilde{O}(n^{2/3})$ *server computation;*
- *the client space is* $\widetilde{O}(n^{2/3})$*.*

Our 2-server result (Theorem 1.5) is different in nature from Beimel et al. [BIM00]. Beimel et al.'s schemes adopt the public-preprocessing model — they require a polynomial amount of server space, but the client is stateless. Our scheme adopts client-specific preprocessing, where the server stores only the original database, but the client now stores a sublinear-sized hint. Table 1 provides a more detailed comparison of the costs.

Motivation: distributing the client. In addition to the fundamental theoretical interest in maximizing security and minimizing assumptions, information-theoretic PIR protocols are particularly useful for applications that distribute the PIR client among two or more parties. For instance, this is the case in protocols for secure multiparty computation (MPC) that access secret locations in a public or secret-shared database. While one can use general MPC techniques to distribute the role of the client in previous OWF-based protocols [ZPSZ24, MIR23, GZS24a], the resulting MPC protocol will make a *non-black-box* use of the underlying OWF, which typically incurs a big concrete overhead. In contrast, an information-theoretic protocol can be distributed while only making a black-box use of cryptography, or even without cryptography in an honest-majority setting. In fact, this served as a primary motivation for studying information-theoretic variants of related primitives, such as ORAM [DMN11] and distributed point functions [BGIK22]. The distributed variants of our protocols enjoy their qualitative efficiency features of sublinear space and per-query time without requiring non-black-box use of cryptography.

1.2 Technical Highlights

New lower bound techniques. We now explain the intuition behind the proof of our lower bound, i.e., Theorem 1.1. Given a preprocessing PIR scheme with less than $t/2$ client space and less than $n/2$ total bandwidth across t consecutive queries, we construct a mutual information amplification (MIA) protocol, and we show that MIA implies one-way functions (OWF). An MIA protocol is similar to a key exchange protocol in that it allows Alice and Bob to establish a shared key. However, in an MIA protocol, Alice and Bob are allowed to share m bits of mutual information upfront. The goal is to agree on a key that computationally appears to have ℓ bits of Shannon entropy conditioned on the protocol transcript, where $\ell > m$.

We construct an MIA protocol from PIR as follows. Initially, Alice is given a random n-bit (preprocessed) database, and Bob is given the hint of size $< t/2$. Therefore, they share $m < t/2$ bits of mutual information. Next, Bob picks a random chunk of t consecutive positions $i_1, \ldots, i_t$ and sends the indices to Alice. Alice and Bob now run the PIR protocol, at the end of which Bob learns the database's values at these t positions. They both output the database value at these positions as the shared key $\mathsf{k} = (\mathbf{DB}[i_1], \ldots, \mathbf{DB}[i_t])$. An adversary cannot distinguish this protocol execution from one where the PIR is run on completely unrelated positions $i'_1, \ldots, i'_t$ that are independent of the indices $i_1, \ldots, i_t$. But in that case the key is guaranteed to have high Shannon entropy conditioned on the view of the protocol execution. This is because, conditioned on the view of a protocol with communication complexity $\leq n/2$, the database has entropy-rate $\geq 1/2$, and therefore so does the random/independent t-bit chunk of the database, meaning that the conditional entropy of k is $\ell \geq t/2$. Finally, we show that given a MIA scheme that is able to expand the parties' mutual information, the protocol transcript together with the shared key at the end have noticeable false entropy, which implies OWF by [IL89, Lemma 4.5].

SZK barrier. A natural question is whether OWFs, or even stronger "symmetric" primitives, are sufficient to circumvent the above lower bound. We give evidence against this for "database-oblivious" schemes in which the client's questions are independent of the database-dependent responses of the server (including any database-dependent part of the hint). Concretely, we show that any database-oblivious scheme that circumvents the lower bound implies a hard promise problem in the complexity class SZK. We start by observing that any such database-oblivious scheme with t online queries implies a one-round PIR scheme in which an offline hint of size $|h| \ll t$ can be used to privately retrieve a single record of length t with $\ll n$ download bandwidth. We then show that if we replace the hint by having the server apply a pairwise-independent hash function to each record, with output length $\approx |h|$, then a computationally unbounded client can still decode the chosen record (with negligible failure probability) by just enumerating over all possible hints and checking the hash value. Moreover, since the hash function shrinks each record, this change keeps the download smaller than the database size. The resulting scheme can be viewed as a single-server PIR scheme with small error probability and a computationally unbounded decoding function, or alternatively as a relaxed variant of an SSB hash function [HW15]. The last step of our proof shows that this primitive implies an average-case hard promise problem in SZK. This step closely follows a similar result for standard PIR schemes which is implicit in [LV16]. The final observation is that all of the above steps relativize, implying that an unconditional ROM construction would separate (promise) SZK from BPP relative to a random oracle, settling an open question in complexity theory.

New upper bound techniques. To get Theorem 1.4, we introduce two novel ideas which may be of independent interest. First, we devise a non-trivial method to combine core ideas from the more recent "puncturable pseudorandom set" paradigm first proposed by Corrigan-Gibbs and Kogan [CK20], as well as classical techniques originally described by the original work of Chor et al. [CGKS95]. Previously, it was unknown how to combine Corrigan-Gibbs and Kogan's new paradigm [CK20] (which was also adopted in a flurry of more recent works [SACM21, CHK22, ZLTS23, LP22]) with the rich techniques developed in the classical PIR literature. In this sense, our work unveils some interesting connections between the two bodies of literature.

Second, we propose a novel technique for achieving adaptive correctness. In comparison, some recent works in the preprocessing PIR literature [ZPSZ24, GZS24a] assume that the client's queries are chosen independently of the adversary's view so far in the protocol, and this assumption is needed for correctness hold — see Section 1.3 for an explicit adaptive correctness attack against these schemes. This non-adaptive assumption on the adversary is similar in nature a known-plaintext adversary in the context of encryption, which is generally considered too weak in many practical applications. To get Theorem 1.4, we also first construct a scheme with non-adaptive correctness as a stepping stone. However, we then show how to upgrade the scheme to achieve adaptive correctness, while preserving the asymptotical performance. The ideas behind this upgrade can be of independent interest: while it is outside the scope of our paper, we believe that our framework can be used to upgrade existing schemes [ZPSZ24, GZS24a] to achieve adaptive correctness.

1.3 Additional Discussions on Related Work

Additional related work. Table 1 focuses on comparing information-theoretic constructions with sublinear computation. We now survey the broader literature. Computationally secure single-server PIR with $\widetilde{O}(1)$ bandwidth is known in the classical setting with linear computation per query, relying on various assumptions such as LWE, ϕ-hiding, decisional composite residuosity (DCR), and linear homomorphic encryption [HHCG+23, MW22, ACLS18, CMS99, Lip09, Lip05]. More recently, the ground-breaking work of Lin, Mook, and Wichs [LMW23] showed a PIR scheme in the global preprocessing model that achieves polylogarithmic bandwidth and communication per query assuming Ring LWE. All of these works require cryptographic primitives that imply public-key cryptography.

Our work should also be distinguished from some earlier works that claim to achieve information-theoretic security but works only for one query (e.g., Theorem 11 of Corrigan-Gibbs and Kogan [CK20]). Unless otherwise stated, our work focuses on the multi-query setting because we want to amortize the cost of the preprocessing over unbounded number of queries.

Explicit adaptive correctness attacks on some prior schemes. We explain why earlier schemes including Piano [ZPSZ24] and Goshal et al. [GZS24a] are explicitly broken under an adaptive correctness attack. The attack exploits their reliance on the following load balancing strategy: except with negligible probability, in every window of $\widetilde{O}(\sqrt{n})$ queries, the number of queries that land in each chunk is at most $\widetilde{O}(1)$. In particular, preprocessing is performed every window of $\widetilde{O}(\sqrt{n})$ queries, and this preprocessing provisions only $\widetilde{O}(1)$ replacement entries for each chunk. Now each query lands in some chunk and consumes a replacement entry of the chunk. To achieve load balancing, we want each query to land in a random chunk. In Piano [ZPSZ24] and Goshal et al. [GZS24a], this is achieved by randomly permuting the database indices using a public PRP upfront, and assuming that queries are chosen independently of the PRP. Therefore, the adaptive correctness attack is: after observing the PRP, simply choose $\sqrt{n}$ queries that all land in one chunk.

2 Definitions

A k-server, preprocessing Private Information Retrieval (PIR) scheme consists of the following stateful protocols:

- **Preprocessing**. The preprocessing phase is executed once upfront. Each server's input is a database denoted $\mathbf{DB} \in \{0,1\}^n$. The client has no input. The client and the server(s) runs a preprocessing protocol at the end of which each party may store some local state.

- **Query**. After the preprocessing phase, the query phase can be repeated multiple times. The client's input is an index $id \in \{0, 1, \dots, n-1\}$. The servers have no input. The (stateful) client and server(s) run a query protocol at the end of which the client outputs answer, and the servers have no output.

If the maximum number of queries $T(n)$ is known upfront, we call it the *bounded* query setting. Otherwise, if the maximum number of queries $T(n)$ is unknown upfront, we call it the *unbounded* query setting. Throughout the paper, we assume that the total number of queries $T(n)$ is upper bounded by some polynomial function in n.

Perfect privacy. We say that a k-server preprocessing PIR scheme satisfies perfect privacy, iff there exists a simulator Sim, such that for any n, for any $\mathbf{DB} \in \{0,1\}^n$, for any sequence of queries $id_1, id_2, \dots, id_{T(n)} \in \{0, 1, \dots, n-1\}$, for any $j \in [k]$, the following two distributions are identically distributed:

- *Real execution.* Run the real world PIR protocol where the server's input database is $\mathbf{DB}$, and the client's queries are $id_1, id_2, \dots, id_{T(n)}$, output the j-th server's view during the protocol.
- *Ideal execution.* Output $\mathsf{Sim}(n, \mathbf{DB}, T(n), j)$.

Since the simulator Sim is not given the sequence of queries $id_1, id_2, \dots, id_{T(n)}$, the above privacy definition captures the intuition that any single server's view leaks nothing about the client's queries.

Note that in this paper, we focus on the single-server and two-server settings. Therefore, in the above privacy definition, we assume that the adversary has control of only one server. For the case of more general $k > 2$, it is also meaningful to consider a t-out-of-k style definition where the adversary may control up to t servers.

In our paper, all the constructions have the following additional features: 1) the client makes a *single streaming pass* over the database during preprocessing; 2) each query can be accomplished with a *single round-trip*; 3) the server only needs to store the original database and *no additional state*; and 4) the schemes are data-oblivious. However, our lower bound proof (Section 5) works even for protocols with multiple rounds, and when the server is allowed to store an arbitrary amount of additional state beyond just the database itself.

Non-adaptive correctness. We say that a preprocessing PIR scheme satisfies non-adaptive correctness, iff for any $T(n)$ that is upper bounded by some polynomial function in n, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for any n, any[3] $\mathbf{DB} \in \{0,1\}^n$, any sequence of $T(n)$ queries denoted $id_1, \ldots, id_{T(n)} \in \{0,1,\ldots,n-1\}$, with $1-\mathsf{negl}(n)$ probability, the client outputs correct answers for all $T(n)$ queries after executing the honest protocol with the servers.

The above non-adaptive correctness definition is the same the one used in some earlier works [ZPSZ24, SACM21, GZS24b].

Adaptive correctness. Consider the following execution $\mathsf{AdaptiveCorrect}^{\mathsf{DB},\mathcal{A},n,T(n)}$ where the client interacts with an adversary $\mathcal{A}$ who controls t out of the k servers.

Experiment $\mathsf{AdaptiveCorrect}^{\mathsf{DB},\mathcal{A},n}$:

- Throughout the execution, the t servers controlled by the adversary $\mathcal{A}$ always follows the honest protocol.
- First, the client and the servers run the preprocessing protocol where all servers receive the input $\mathbf{DB}$;
- Next, the adversary $\mathcal{A}(n, \mathbf{DB})$ repeats the following: each time the adversary $\mathcal{A}$ adaptively chooses a next query $id \in \{0,1,\ldots,n-1\}$ for the client. The client and the servers now run the query protocol.
- Finally, when the adversary $\mathcal{A}$ stops, if the client has output the correct answers for all queries, then output 1.

We say that a preprocessing PIR scheme satisfies adaptive correctness iff for any $\mathcal{A}$ which repeats the query phase polynomially in n many times, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for any n, and any $\mathbf{DB} \in \{0,1,\ldots,n\}^n$, the above experiment $\mathsf{AdaptiveCorrect}^{\mathsf{DB},\mathcal{A},n}$ outputs 1 with probability $1-\mathsf{negl}(n)$.

Intuitively, the difference between non-adaptive and adaptive correctness is the following. For non-adaptive correctness, we implicitly assume that the clients' queries are chosen independently of the randomness consumed by the PIR scheme itself. For adaptive correctness, we require that correctness still hold even when the adversary can adaptively choose the queries in a way that depends on its view in the protocol so far. In some applications of PIR (e.g., private DNS, private web search, private contact discovery), the queries are generated by a human independently of the PIR schemes' random coins. In these cases, it may be reasonable to use the weaker non-adaptive correctness notion. In other applications, adaptive correctness may be necessary, especially if the PIR scheme is used as a building block in a bigger protocol, and for compositional reasons we may need to consider the case when the adversary can influence the choice of the queries of the honest client.

[3]Rather than quantifying over all $\mathbf{DB}$, we can alternatively let the adversary $\mathcal{A}$ choose the $\mathbf{DB}$ upfront. The two definitional approaches are equivalent.

3 Naïve Constructions of Information Theoretic, Single-Server Preprocessing PIR

3.1 Construction from 2-Server Linear PIR with Nearly Optimal Client Space and Bandwidth Tradeoff

If we did not care about the server's computational overhead, we can construct an information-theoretic single-server preprocessing PIR from any *linear*, 1-round, information-theoretic, classical 2-server PIR scheme (that satisfies some natural properties). Here, the "linear" property requires that the answer from either server is a linear combination of the database's values. Further, using the state-of-the-art classical 2-server PIR [DG16] as the underlying PIR, we can get an information-theoretic single-server preprocessing PIR with $n^{1/2+o(1)}$ client space, and $O(\sqrt{n})$ bandwidth per query — in light of the lower bound in Section 5 which holds even when allowing unbounded server computation, the client space/bandwidth tradeoff is tight up to sub-polynomial factors. This is also why the resulting construction is interesting from a theoretical perspective.

Henceforth, we use a tuple of possibly randomized algorithms $(\mathsf{query}, \mathsf{ans}_1, \mathsf{ans}_2, \mathsf{dec})$ to denote a 2-server PIR:

- $(\mathsf{msg}, \mathsf{msg}', st) \leftarrow \mathsf{query}(n, q)$: takes n and the client's query $q \in \{0, 1, \ldots, n-1\}$ as input and outputs two messages, one intended for each server, as well as some state st;
- $\mathsf{res} \leftarrow \mathsf{ans}_1(\mathbf{DB}, \mathsf{msg})$ and $\mathsf{res}' \leftarrow \mathsf{ans}_2(\mathbf{DB}, \mathsf{msg}')$: both functions take in the database and a message received from the client and output a response;
- $\beta \leftarrow \mathsf{dec}(st, \mathsf{res}, \mathsf{res}')$: takes in st as well as a response from each server denoted res and res' respectively, and outputs the query result $\beta \in \{0, 1\}$.

We assume that the scheme satisfies perfect privacy, i.e., any single server's view is independent of the client's query $q \in \{0, 1, \ldots, n-1\}$. We also require the classical 2-server PIR scheme to have a few additional properties which are satisfied by natural 2-server PIR schemes known so far [DG16, CGKS95]:

1. *Linearity:* We can view the database as a vector over some finite ring; further, both ans_1 and ans_2 compute linear combinations of the database's entries over the ring. Specifically, if the output of ans_1 or ans_2 is a vector, then each coordinate is a linear combination of the database's entries.
2. *Obliviously one-sided query generation:* since the 2-server PIR scheme satisfies perfect privacy, each individual coordinate of the query function's output is independent of the client's query. Therefore, we assume that the scheme admits an alternative way for the client to compute its queries to both servers. Specifically, we assume that there exist efficient functions $(\mathsf{query}_1, \mathsf{query}_2)$:
   - $(\mathsf{msg}, st) \leftarrow \mathsf{query}_1(n)$: takes only n as input but not the actual query, and outputs the message to send to the first server, as well as some state st;
   - $(\mathsf{msg}', st') \leftarrow \mathsf{query}_2(st, q)$: takes st and the actual query $q \in \{0, 1, \ldots, n-1\}$ as input, and outputs a message to the second server, as well as internal state st'.

   We want that for any n, any $q \in \{0, 1, \ldots, n-1\}$, let $(\mathsf{msg}, st) \leftarrow \mathsf{query}_1(n)$ and $(\mathsf{msg}', st') \leftarrow \mathsf{query}_2(st, q)$, then the tuple $(\mathsf{msg}, \mathsf{msg}', st')$ is identically distributed as the output of $\mathsf{query}(n, q)$.

Generic construction. We can construct a single-server preprocessing PIR as follows. Specifically, the preprocessing is executed upfront, and during each phase of t queries, the client piggybacks the work of the next phase's preprocessing over this phase's queries.

- **Preprocessing:** *// the next phase's preprocessing is piggybacked over this phase's queries*
  - for each $i \in [t]$, the client runs $(\mathsf{msg}_i, st_i) \leftarrow \mathsf{query}_1(n)$, and initializes a vector $\mathsf{res}_i = \mathbf{0}$ of appropriate length that matches the output length of ans_1;
  - make a streaming pass over the database: for each $\mathbf{DB}[j]$ read, for each hint $i \in [t]$, suppose $\mathsf{ans}_1(\mathbf{DB}, \mathsf{msg}_i) = \sum_j \mathbf{c}_j \mathbf{DB}[j]$, then let $\mathsf{res}_i := \mathsf{res}_i + \mathbf{c}_j \cdot \mathbf{DB}[j]$. Note that the client only needs to compute the coefficient vector $\mathbf{c}_j$ and need not compute the other coefficients for performing this update.
  - Let the hint table be $\{\mathsf{msg}_i, st_i, \mathsf{res}_i\}_{i \in [t]}$.
- **Query:** upon receiving an index $q \in \{0, 1, \ldots, n-1\}$, the client grabs the next unconsumed hint from the hint table of the form $(\mathsf{msg}, st, \mathsf{res})$, and computes $(\mathsf{msg}', st') \leftarrow \mathsf{query}_2(st, q)$. It then sends msg' to the server, and obtains the answer res'. Finally, output $\mathsf{dec}(st', \mathsf{res}, \mathsf{res}')$.

The perfect privacy and adaptive correctness properties of the above scheme are inherited by those of the underlying classical 2-server PIR scheme. Specifically, for privacy, note that the server observes a single streaming pass over the database during preprocessing, which does not leak any information. The view of the server for the queries is identially distributed as the view of one server in the underlying 2-server PIR.

Instantiation from Dvir-Gopi's 2-server PIR. The most efficient classical, information-theoretic 2-server PIR known is the scheme by Dvir and Gopi [DG16]. It is not hard to check that their scheme satisfies the natural properties that we need. Specifically, the query algorithm simply chooses a random vector $\mathbf{z}$ of length $k = \exp(O(\sqrt{\log n \log\log n}))$ from $\mathbb{Z}_6^k$. The query to the first server is simply $\mathbf{z}$ which can be computed obliviously without knowing the query, and the query to the second server can be computed from $\mathbf{z}$ and the query $q \in \{0, 1, \ldots, n-1\}$. Therefore, each hint can be described in $O(k)$ size. During the streaming pass in the preprocessing phase, the client can compute the coefficient vector $\mathbf{c}_j \in \mathbb{Z}_6^k$ corresponding to each database position $\mathbf{DB}[j]$ in $O(k)$ time. Therefore, the client's total computation during preprocessing is $O(n \cdot t \cdot k)$. Amortizing the work over t queries, the client computation per query is $O(n \cdot k)$. The server computation per query is also $O(n \cdot k)$ which is the same as the server computation of the original Dvir-Gopi PIR [DG16].

Summarizing the above, we conclude with the following theorem:

Theorem 3.1 (Achieving nearly optimal bandwidth/client-space tradeoff)**.** *There exists an information-theoretic, perfectly private, and adaptively correct single-server preprocessing PIR scheme supporting unbounded, arbitrary queries, and achieving the following performance bounds:*

- *there is an initial preprocessing phase in which the client makes a streaming pass over the database consuming $O(n)$ bandwidth and server computation; and the client performs $O(n \cdot t \cdot \exp(\sqrt{\log n \log\log n}))$ computation;*
- *after the initial preprocessing, every query takes $O(\exp(\sqrt{\log n \log\log n}))$* online *bandwidth, $O(n/t)$* offline *bandwidth, and $O(n \cdot \exp(\sqrt{\log n \log\log n}))$ client and server computation;*
- *the client space is $O(t \cdot \exp(\sqrt{\log n \log\log n}))$.*

3.2 Construction from Recent PRF-Based Constructions

Assuming the existence of pseudorandom functions (PRF), Piano [ZPSZ24] and the subsequent work of Mughees et al. [MIR23] constructed a single-server preprocessing PIR scheme with $O(\sqrt{n})$ bandwidth, and $\widetilde{O}(\sqrt{n})$ server and client computation per query, and consuming $\widetilde{O}(\sqrt{n})$ client space. In their schemes, the only use of the PRF is to compress the client space. Specifically, in these schemes [ZPSZ24, MIR23], every hint corresponds to a random set S of size $O(\sqrt{n})$ and the parity $\oplus_{i \in S} \mathbf{DB}[i]$; moreover, the client stores $\widetilde{O}(\sqrt{n})$ hints in total.

A straightforward but inefficient way to make their schemes information theoretic is to remove the PRF, and have each hint directly store the expanded random set of size $O(\sqrt{n})$. However, this would result in $\widetilde{O}(n)$ client space.

A balancing trick. One key observation is that the space overhead for the storing the expanded set and the parity bit is unbalanced: storing the expanded set requires $\sqrt{n}$ space but the parity bit requires only 1 bit. We can use a balancing trick to balance these two overheads, by making the block size $n^{1/3}$. In this way, a database of n bits can be viewed as a database with $n^{2/3}$ blocks where each block has $n^{1/3}$ bits. For the client to fetch any bit, it has to fetch the entire block that the bit resides in. Therefore, the idea is to apply Mughees et al. [MIR23] over a database of $n^{2/3}$ blocks each of $n^{1/3}$ bits, and have the client store the expanded sets in the hints. In this case, each phase has $n^{1/3}$ queries, and the client stores $\widetilde{O}(n^{1/3})$ hints. As before, the client piggybacks the next phase's preprocessing over the current phase's queries. In each hint, storing the random set and storing the parity block both take $O(n^{1/3})$ space. Similarly, during each query, the client uploads two random sets of size $O(n^{1/3})$ to the server, and the serve responds with two parity blocks of size $O(n^{1/3})$ — thus the upload and download bandwidth are now also balanced. Summarizing the above, we get the following performance overhead as stated in Theorem 3.2:

Adaptive correctness and privacy of the construction follow directly from the adaptive correctness and privacy of Mughees et al. [MIR23]. Note that we can also use Piano [ZPSZ24] as the underlying scheme instead of Mughees et al. [MIR23], but the resulting scheme would only satisfy non-adaptive correctness since Piano satisfies only non-adaptive correctness. By contrast, Mughees et al. [MIR23] introduced an elegant technique to achieve adaptive correctness.

Theorem 3.2 (Information theoretic single-server preprocessing PIR from Mughees et al. [MIR23])**.** *There exists an information-theoretic, perfectly private, and adaptively correct single-server preprocessing PIR scheme supporting unbounded, arbitrary queries, and achieving the following performance bounds:*

- *there is an initial preprocessing phase in which the client makes a streaming pass over the database consuming $O(n)$ bandwidth and server computation, and the client performs $\widetilde{O}(n)$ computation;*
- *after the initial preprocessing, every query takes $O(n^{2/3})$ online and offline bandwidth, $\widetilde{O}(n^{2/3})$ client computation, and $O(n^{2/3})$ server computation;*
- *the client space is $\widetilde{O}(n^{2/3})$.*

4 Warmup: Scheme for Bounded, Random, and Distinct Queries

4.1 Set Definitions and Operations

Without loss of generality, we may assume that $n^{1/6}$ is an integer since otherwise we can always pad n to a 6-th power. Henceforth let $m = n^{1/3}$. We will express an index from $id \in \{0, 1, \ldots, n-1\}$ as a tuple $id := (x, y, z) \in \{0, 1, \ldots, m-1\}^3$. In other words, (x, y, z) is the base-m representation of id. We often write $\mathbf{DB}[x, y, z] = \mathbf{DB}[id]$ if $id = (x, y, z)$, and use these notations interchangeably to index into the database.

Set distribution $\mathcal{D}$. We define the distribution $\mathcal{D}$ which samples a random set $S \subseteq \{0, 1, \ldots, n-1\}$ of expected size $\sqrt{n}$. Specifically,

1. for each $x \in \{0, 1, \ldots, m-1\}$, sample x with probability $\frac{1}{n^{1/6}}$, and let X be the set of x-indices sampled in this way;
2. for each $y \in \{0, 1, \ldots, m-1\}$, sample y with probability $\frac{1}{n^{1/6}}$, and let Y be the set of y-indices sampled in this way;
3. for each $z \in \{0, 1, \ldots, m-1\}$, sample z with probability $\frac{1}{n^{1/6}}$, and let Z be the set of z-indices sampled in this way.
4. Finally, output the cross-product $X \times Y \times Z$.

Fact 4.1. *We immediately have the following facts about $\mathcal{D}$ where $\mathsf{negl}(\cdot)$ denotes a suitable negligible function:*

- **Size of each dimension.** $\mathbb{E}[|X|] = \mathbb{E}[|Y|] = \mathbb{E}[|Z|] = m/n^{1/6} = n^{1/3}/n^{1/6} = n^{1/6}$. *Further, except with $\mathsf{negl}(n)$ probability, the sizes of X, Y, Z are between $0.9n^{1/6}$ and $1.1n^{1/6}$.*
- **Set size**. *The probability that each $(x, y, z) \in \{0, \ldots, m-1\}^3$ is contained in S is $(1/n^{1/6})^3 = 1/\sqrt{n}$. Therefore, the expected set size $\mathbb{E}[|S|] = \sqrt{n}$. Further, except with $\mathsf{negl}(n)$ probability, the size of S is between $0.6\sqrt{n}$ and $1.5\sqrt{n}$.*
- **Set description size.** *S can be described with X and Y and Z. Therefore, the expected description size of S is $3 \cdot n^{1/6}$. Further, except with $\mathsf{negl}(n)$ probability, $|\mathsf{desc}(S)|$ is between $2.7n^{1/6}$ and $3.3n^{1/6}$.*

Resample operation. Given a set S such that $(x, y, z) \in S$, we define the resampling operation denoted $\mathsf{Resample}(S, (x, y, z))$ to output the following set:

- write S as X, Y, and Z; it is known that $x \in X$, $y \in Y$, and $z \in Z$;
- resample the decisions whether $x \in X$, $y \in Y$, and $z \in Z$; specifically, the probabilities that $x \in Y$ $y \in Y$, or $z \in Z$ should all be $\frac{1}{n^{1/6}}$; let X', Y', and Z' be the resampled outcomes;
- output $X' \times Y' \times Z'$.

Parity of a set. Given a set $S = X \times Y \times Z$, and some database $\mathbf{DB} = \{0, 1\}^n$, we define the parity of the set S as

$$\mathsf{parity}(S) = \oplus_{(x,y,z) \in S} \mathbf{DB}[x, y, z]$$

Reconstruction algorithm. Let $S = X \times Y \times Z$ and let $(x, y, z) \in S$. Suppose we are given the parities of the following sets:

- $p_{111} = \mathsf{parity}(X \times Y \times Z)$,
- $p_{011} = \mathsf{parity}\left((X \backslash \{x\}) \times Y \times Z\right)$,
- $p_{101} = \mathsf{parity}\left(X \times (Y \backslash \{y\}) \times Z\right)$,

- $p_{110} = \mathsf{parity}\,(X \times Y \times (Z \backslash \{z\}))$,
- $p_{001} = \mathsf{parity}\,((X \backslash \{x\}) \times (Y \backslash \{y\}) \times Z)$,
- $p_{010} = \mathsf{parity}\,((X \backslash \{x\}) \times Y \times (Z \backslash \{z\}))$,
- $p_{100} = \mathsf{parity}\,(X \times (Y \backslash \{y\}) \times (Z \backslash \{z\}))$,
- $p_{000} = \mathsf{parity}\,((X \backslash \{x\}) \times (Y \backslash \{y\}) \times (Z \backslash \{z\}))$,

we can reconstruct $\mathbf{DB}[x, y, z]$ in the following way:

$$\mathbf{DB}[x, y, z] = \oplus_{i,j,k \in \{0,1\}} p_{i,j,k}$$

To see this, observe that except for (x, y, z), every other element appears an even number of times in the final xor-sum.

4.2 PIR Scheme for $Q = \sqrt{n}/10$ Random Distinct Queries

We first describe a PIR scheme which achieves perfect privacy regardless of the queries, and achieves correctness with probability at least 0.1 as long as there are only $Q = \sqrt{n}/10$ random and distinct queries. It is easy to amplify the correctness probability to $1 - \mathsf{negl}(n)$ by running $\omega(\log n)$ independent instances in parallel. Therefore, henceforth, we focus on describing the algorithm for a single instance. Further, in Appendix A, we show how to remove the restriction of having only bounded, random, and distinct queries, such that we can finally support unbounded, arbitrary queries.

Without loss of generality, we assume that $n \geq \lambda$ since we care about the asymptotical performance of the scheme when n is large.

Preprocessing phase.

1. The client samples $L = 20\sqrt{n}$ random sets denoted $S_1, \ldots, S_L$.
2. For each $i \in [L]$, write $S_i = X_i \times Y_i \times Z_i$, the client wants to store the parities of the following sets:
   - $S_i = X_i \times Y_i \times Z_i$;
   - for each $x \in X_i$: $((X_i \backslash \{x\}) \times Y_i \times Z_i)$;
   - for each $y \in Y_i$: $(X_i \times (Y_i \backslash \{y\}) \times Z_i)$;
   - for each $z \in Z_i$: $(X_i \times Y_i \times (Z_i \backslash \{z\}))$;

The client can compute all parities by making a single streaming pass over the database while consuming only $O(n^{2/3})$ space.

Henceforth, we refer to each set S_i along with the associated parities a *hint*, and we refer to the union of all L hints the client's *hint table*.

Queries. We describe how to support $\sqrt{n}$ distinct, random queries. Henceforth let (x, y, z) be the current query.

1. *Client*: find in its hint table a hint with the set S such that the queried index $(x, y, z) \in S$. If there are multiple such hints, choose one at random.

   If no such hints are found, then sample a fresh set S from the distribution $\mathcal{D}$, send S to the server, ignore the server's responses, and output 0. Otherwise, continue with the following.

2. *Client*: let $S' \leftarrow \mathsf{Resample}(S, (x, y, z))$ and send (a succinct description of) S' to the server.
3. *Server*: write $S' = X' \times Y' \times Z'$, and send the parities of the following sets back to the client:
   - $S' = X' \times Y' \times Z'$;
   - for each $x \in \{0, 1, \ldots, m-1\}$, $(X' \cup \{x\}) \times Y' \times Z'$;
   - for each $y \in \{0, 1, \ldots, m-1\}$, $X' \times (Y' \cup \{y\}) \times Z'$;
   - for each $z \in \{0, 1, \ldots, m-1\}$, $X' \times Y' \times (Z' \cup \{z\})$;
4. *Client*: define the good event G to be the event that the set S consumed corresponds to a good hint (i.e., not broken) in the hint table, and moreover, the earlier $S' \leftarrow \mathsf{Resample}(S = X \times Y \times Z, (x, y, z))$ operation successfully removed all of x, y, and z from X, Y, and Z respectively, do the following:
   - If the good event G happens, then, the client can obtain the parities $p_{111}, p_{011}, p_{101}, p_{110}$ from the parities associated with the set S stored in its hint table (see Section 4.1 for the definition of these parities when given S and (x, y, z)). Further, the client can obtain the parities $p_{000}, p_{100}, p_{010}, p_{001}$ from the server's response. The client calls the reconstruction algorithm and outputs the answer.
   - Else if the good event G does not happen, simply output 0 as the answer.
5. *Client*: Sample a new set S according to the distribution $\mathcal{D}$ subject to containing (x, y, z), and replace the consumed hint in the hint table with the newly sampled set. Mark this hint as *broken*.

Additional details of the preprocessing algorithm. We give more details on the preprocessing algorithm to show that the client can accomplish it using only $O(n^{2/3})$ space and $O(n)$ computation.

During the streaming pass, for each hint of the form $S_i = X_i \times Y_i \times Z_i$, the client updates the cumulative parities for the following *planar sets*: $\{\{x\} \times Y_i \times Z_i\}_{x \in X_i}$, $\{X_i \times \{y\} \times Z_i\}_{y \in Y_i}$, and $\{X_i \times Y_i \times \{z\}\}_{z \in Z_i}$. Each planar set has size $n^{1/3}$, and each hint has $n^{1/6}$ planar sets.

The client proceeds in batches, and for each batch, the client downloads the next $n^{2/3}$ bits from the server — we may use $\mathbf{DB}[x, *, *]$ to denote the batch where $*$ denotes wildcard. Now, the client performs the following. For each hint of the form $X_i \times Y_i \times Z_i$, if $x \in X_i$, then for each $(y, z) \in Y_i \times Z_i$, the client accumulates $\mathbf{DB}[x, y, z]$ into the parities of the following planar sets associated with the hint: $\{x\} \times Y_i \times Z_i$, $X_i \times \{y\} \times Z_i$, and $X_i \times Y_i \times \{z\}$.

Finally, for each hint of the form $X_i \times Y_i \times Z_i$, the client computes the parities of the following sets: $X_i \times Y_i \times Z_i$, $\{(X_i \backslash \{x\}) \times Y_i \times Z_i\}_{x \in X_i}$, $\{X_i \times (Y_i \backslash \{y\}) \times Z_i\}_{y \in Y_i}$, and $\{X_i \times Y_i \times (Z_i \backslash \{z\})\}_{z \in Z_i}$ from the parities of the planar sets associated with the hint.

4.3 Performance Bounds

Henceforth, we ignore the negligible probability that Fact 4.1 does not hold. In other words, all the performance bounds stated below hold with all but negligible probability.

Preprocessing phase. We first analyze the cost of preprocessing.

- *Bandwidth and server work.* The client makes a single pass over the database, consuming $O(n)$ bandwidth. Since the server does not do any additional work besides streaming the database to the client, its work is also $O(n)$.

- *Client space.* For each of the $L = O(\sqrt{n})$ sets, the client needs to store 1) the description of the set which is $O(n^{1/6})$ in size; and 2) $O(n^{1/6})$ parities. Therefore, the total client space is $O(L \cdot n^{1/6}) = O(n^{2/3})$. Further, the above preprocessing algorithm requires only $O(n^{2/3})$ client space as well, since at any time, the client only needs to store the current batch of size $n^{2/3}$, and the parities of $O(L \cdot n^{1/6}) = O(n^{2/3})$ sets.
- *Client work.* For each batch of size $n^{2/3}$, the client scans through all hints which takes $O(n^{1/2})$ work. Further, for each (x, y, z) in the current batch, $O(1)$ number of planar set updates are needed for each hint that contains (x, y, z). Because each index (x, y, z) is contained in $O(1)$ hints in expectation, the total update work is $O(n^{2/3})$ except with $\mathsf{negl}(n)$ probability. Therefore, for each batch, the expected total work is upper bounded by $O(n^{1/2} + n^{2/3}) = O(n^{2/3})$. Summing over all $n^{1/3}$ batches, the work so far is $O(n)$. Finally, each hint takes $O(n^{1/6})$ time to reconstruct the parities of the sets $X_i \times Y_i \times Z_i$, $\{(X_i \backslash \{x\}) \times Y_i \times Z_i\}_{x \in X_i}$, $\{X_i \times (Y_i \backslash \{y\}) \times Z_i\}_{y \in Y_i}$, and $\{X_i \times Y_i \times (Z_i \backslash \{z\})\}_{z \in Z_i}$ from the parities of the planar sets. This step takes $O(n^{1/2} \cdot n^{1/6}) = O(n^{2/3})$ work. Therefore, the total client work during preprocessing is $O(n)$.

Each query. We now analyze the cost of each query.

- *Bandwidth.* For each query, the client sends a resampled set S' to the server. Since the description size of S' is $O(n^{1/6})$, the upload bandwidth required is $O(n^{1/6})$. Further, the server sends the client $O(n^{1/3})$ parities which consumes $O(n^{1/3})$ download bandwidth.
- *Client work.* The client needs to scan the $L = O(\sqrt{n})$ hints in its hint table and find the hints that contain the query (x, y, z). Suppose for each hint $S = X \times Y \times Z$ we store the sets X, Y, and Z using a Cuckoo hash table (which does not asymptotically increase the space consumption), we can check whether $(x, y, z) \in S$ in constant time. Therefore, this step takes $O(\sqrt{n})$ work. The client receives a response of size $O(n^{1/3})$ from the server which takes $O(n^{1/3})$ work. Further, the client needs to perform the $\mathsf{Resample}$ operation, reconstruct $\mathbf{DB}[x, y, z]$ after receiving the response from the server, and sample a new set subject to containing (x, y, z). The work of these steps is dominated by others. Specifically, sampling a set subject to containing (x, y, z) can be achieved by flipping a coin for every $x' \neq x$ to decide if x' should be chosen, and forcing x to be chosen and similarly for the other two dimensions. Therefore, the total client work is $O(\sqrt{n})$.
- *Server work.* When the server receives $S' = X' \times Y' \times Z'$, it needs to compute $O(n^{1/3})$ parities as the response to the client. To compute the response, the server can compute the parity of $S' = X' \times Y' \times Z'$, as well as the parities of the following planar sets: $\{\{x\} \times Y' \times Z'\}_{x \in \{0,\ldots,m-1\}}$, $\{X' \times \{y\} \times Z'\}_{y \in \{0,\ldots,m-1\}}$, and $\{X' \times Y' \times \{z\}\}_{z \in \{0,\ldots,m-1\}}$. Then, it can compute each of the $O(n^{1/3})$ final parities in constant time. It takes $O(\sqrt{n})$ work to compute the parity of $S' = X' \times Y' \times Z'$, and $O((n^{1/6})^2 \cdot n^{1/3}) = O(n^{2/3})$ work to compute all the parities of all planar sets. Therefore, the total server work is upper bounded by $O(n^{2/3})$.

Amortized cost per query. When we amortize the cost of the preprocessing over each of the $\sqrt{n}$ queries, we get the amortized cost per query:

- *Amortized bandwidth.* The amortized bandwidth per query is $O(n/\sqrt{n}) + O(n^{1/3}) = O(\sqrt{n})$.
- *Amortized client work.* The amortized client work per query is $O(n/\sqrt{n}) + O(\sqrt{n}) = O(\sqrt{n})$.
- *Amortized server work.* The amortized server work per query is $O(n/\sqrt{n}) + O(n^{2/3}) = O(n^{2/3})$.

Finally, the above is the performance of a single instance. Recall that in our final construction, we use super-logarithmically many instances to boost the correctness probability to $1 - \mathsf{negl}(n)$, incurring an extra super-logarithmic factor in cost. However, during the preprocessing phase, all super-logarithmically many copies can share the same streaming pass, so the bandwidth and server computation in the preprocessing phase need not suffer from the superlogarithmic factor.

4.4 Privacy Proof

We construct the following simulator. For the preprocessing phase, simply make a linear pass over the database. For each query, sample a set according to the distribution $\mathcal{D}$ and send it to the adversary. We want to prove that for an arbitrary sequence of queries, the simulated view is identically distributed as the view of the adversary in the real world.

To prove this, we will prove that at the beginning of each query, even when conditioned on the adversary's view so far, 1) the sets in the hint table are distributed as L sets freshly and independently sampled from the distribution $\mathcal{D}$; and 2) the query sent to the adversary is identically distributed as a freshly sampled set from the distribution $\mathcal{D}$. We can prove this inductively. Suppose at the beginning of the t-th query, the hint table is distributed as L sets independently sampled from $\mathcal{D}$. We want to prove that 1) the hint table enjoys the same distribution after the t-th query; and 2) the t-th query itself is distributed as a set sampled from $\mathcal{D}$. Henceforth let (x, y, z) denote the t-th query.

At the beginning of the t-th query, the hint table contains L sets sampled independently from $\mathcal{D}$ — we can equivalently view the distribution as follows:

- First, sample according to a suitable distribution the indices of the hints that match (x, y, z). Let this matched set be I (which is possibly empty).
- For every hint in I, sample a set according to $\mathcal{D}$ but subject to containing (x, y, z); for every hint not in I, sample a set according to $\mathcal{D}$ subject to not containing (x, y, z).

The distribution of the hint table after the t-th query is almost the same as the above, except that one random hint among the set I is sampled twice according to $\mathcal{D}$ but subject to containing (x, y, z). The new distribution is identical as the distribution right before the t-th query. The distribution of the t-query is the same as the following: sample a set S according to $\mathcal{D}$ but subject to containing (x, y, z), and then output $\mathsf{Resample}(S, (x, y, z))$. Clearly, the resulting distribution is identical as sampling a random set from $\mathcal{D}$ without any constraints. It is also easy to observe that the distribution of the t-th query is independent of the distribution of the hint table after the t-th query.

4.5 Proof of Correctness under Random Queries

We now prove correctness under Q random, distinct queries. Later in Appendix A, we show how to extend the scheme to support unbounded, arbitrary queries with adaptive correctness.

Fix an arbitrary time step $t \in [Q]$ and consider the t-th query: we want to show that the client outputs the correct answer with probability at least 0.1. Although the correctness probability for a single instance is only a small constant, we can easily amplify the correctness probability to $1 - \mathsf{negl}(n)$ by running $\omega(\log n)$ independent instances in parallel. Recall that the client knows whether each instance produced a correct answer. Therefore, as long as one of the instances returns a correct answer, the client can obtain the correct answer.

Henceforth, we focus on analyzing a single instance. Unless the following bad events happen, the client is guaranteed to output a correct answer — henceforth let (x, y, z) be the current query:

- *Failure of finding a good match.* Either the client did not find any hint in the hint table that matches the query (x, y, z), or the client happens to choose a broken hint during the query.
- *Failure of removal during* $\mathsf{Resample}$. The $\mathsf{Resample}(S, (x, y, z))$ operation where $S = X \times Y \times Z$ fails to achieve one of the following: 1) remove x from X, 2) remove y from Y, or 3) remove z from Z.

Below, we first show that the probability of the second bad event is $o(1)$. Then, we focus on bounding the probability of the first bad event, and show that it is at most 0.6.

Probability of $\mathsf{Resample}$ failing to remove. The probability of failing to remove x (or y or z, resp.) from X (or Y or Z resp.) is $1/n^{1/6}$. Therefore, the probability that there exists a dimension for which removal fails is at most $3/n^{1/6}$.

Probability of failing to find a good match. Imagine that an honest client interacts with the adversary. Let T_{end} be the client's hint table *after* the adversary makes Q queries where the last query is denoted (x, y, z) also called the challenge query. Let $T^* \subseteq T_{\text{end}}$ be the hints that contain the challenge query (x, y, z). We can partition T^* as $T^* = T^*_{\text{good}} \cup T^*_{\text{bad}}$ where T^*_{good} and T^*_{bad} denote the good and the broken hints at the end of the Q queries, respectively. It is easy to see that each hint in T^*_{bad} must match at least one of the Q queries — specifically, the last time this hint was consumed, it was replaced with a freshly sampled hint subject to containing the relevant query. On the other hand, each hint in T^*_{good} may or may not match any of the Q queries. We want to show that with probability at least 0.5, $|T^*| > 0$ and $\frac{|T^*_{\text{good}}|}{|T^*|} \geq 0.5$, or alternatively, $\frac{|T^*_{\text{bad}}|}{|T^*|} \leq 0.5$. Let $T^*_{\cap} \subseteq T^*$ be the hints that match at least one of the Q queries. Clearly, $T^*_{\text{bad}} \subseteq T^*_{\cap}$. Therefore, it suffices to show that with probability at most 0.5, $|T^*_{\cap}| \geq 0.5\mathsf{Cnt}^*$.

In the privacy proof (Section 4.4), we showed that conditioned on the adversary's view at the end of the Q queries, T^*_{end} is distributed as a freshly sampled set of hints. We can imagine that T^*_{end} is sampled as follows: first, sample the random variable Cnt^* from an appropriate distribution. Then, sample Cnt^* hints subject to containing (x, y, z), and sample $L - \mathsf{Cnt}^*$ hints subject to not containing (x, y, z), and T^*_{end} is a random permutation of all the sampled hints.

Suppose we sample a random hint table with $L = 20\sqrt{n}$ hints. The probability that each set contains the query (x, y, z) is $1/\sqrt{n}$. Therefore, the probability that no set contains (x, y, z) is $(1 - 1/\sqrt{n})^L = (1 - 1/\sqrt{n})^{20\sqrt{n}} \leq e^{-\frac{1}{\sqrt{n}} \cdot 20\sqrt{n}} \leq 1/e^{20}$. Henceforth, we focus on the case when there are $\mathsf{Cnt}^* > 0$ number of hints that match (x, y, z). The following proof holds when conditioning on some $\mathsf{Cnt}^* > 0$.

Proposition 4.2. *Consider the following random experiment: sample a random set S subject to containing (x, y, z), and sample (x', y', z') at random subject to being different from (x, y, z). The probability that $(x', y', z') \in S$ is at most $1.1/\sqrt{n}$.*

Proof. Consider the following slightly modified experiment: sample a random set S containing subject to (x, y, z), and sample (x', y', z') at random *without needing to be distinct from* (x, y, z). To prove the claim, we can equivalently bound

$$\frac{\Pr[(x', y, z') \in S \text{ and } (x', y', z') \neq (x, y, z)]}{\Pr[(x', y', z') \neq (x, y, z)]}$$

in the above modified experiment. Therefore, it suffices to show that $\Pr[(x', y, z') \in S \leq 1.1/\sqrt{n}$ in the above modified experiment where (x', y', z') is sampled at random without the need to be distinct from (x, y, z). We prove this below.

$$\begin{aligned}\Pr[(x',y',z')\in S] =&\Pr[(x'\neq x)\wedge(y'\neq y)\wedge(z'\neq z)\wedge(x',y',z')\in S]\\&+3\cdot\Pr[(x'=x)\wedge(y'\neq y)\wedge(z'\neq z)\wedge(x',y',z')\in S]\\&+3\cdot\Pr[(x'=x)\wedge(y'=y)\wedge(z'\neq z)\wedge(x',y',z')\in S]\\&+\Pr[(x',y',z')=(x,y,z)]\end{aligned}$$

Moreover,

$$\begin{aligned}&\Pr[(x'\neq x)\wedge(y'\neq y)\wedge(z'\neq z)\wedge(x',y',z')\in S]\\\leq&\Pr[(x',y',z')\in S\big|(x'\neq x)\wedge(y'\neq y)\wedge(z'\neq z)]\\=&(1/n^{1/6})^3=1/\sqrt{n}\end{aligned}$$

$$\begin{aligned}&\Pr[(x'=x)\wedge(y'\neq y)\wedge(z'\neq z)\wedge(x',y',z')\in S]\\=&\Pr[(x',y',z')\in S\big|(x'=x)\wedge(y'\neq y)\wedge(z'\neq z)]\cdot\Pr[(x'=x)\wedge(y'\neq y)\wedge(z'\neq z)]\\\leq&\Pr[(x',y',z')\in S\big|(x'=x)\wedge(y'\neq y)\wedge(z'\neq z)]\cdot\Pr[x'=x]\\=&(1/n^{1/6})^2\cdot(1/n^{1/3})=1/n^{2/3}\end{aligned}$$

$$\begin{aligned}&\Pr[(x'=x)\wedge(y'=y)\wedge(z'\neq z)\wedge(x',y',z')\in S]\\=&\Pr[(x',y',z')\in S\big|(x'=x)\wedge(y'=y)\wedge(z'\neq z)]\cdot\Pr[(x'=x)\wedge(y'=y)\wedge(z'\neq z)]\\\leq&\Pr[(x',y',z')\in S\big|(x'=x)\wedge(y'=y)\wedge(z'\neq z)]\cdot\Pr[x'=x]\cdot\Pr[y'=y]\\=&(1/n^{1/6})\cdot(1/n^{1/3})^2=1/n^{5/6}\end{aligned}$$

Finally, $\Pr[(x',y',z')=(x,y,z)]=(1/n^{1/3})^3=1/n$.

Summarizing the above, we have the following for sufficiently large n.

$$\Pr[(x',y',z')\in S]\leq 1.1/\sqrt{n}$$

□

Let $\mathbf{I}_{ij}$ be an indicator random variable such that $\mathbf{I}_{ij}=1$ if the j-th query is contained in the i-th hint that contains (x,y,z). Proposition 4.2 implies that for fixed i, j, $\mathbb{E}[\mathbf{I}_{ij}]\leq 1.1/\sqrt{n}$. We have that

$$\mathbb{E}[|T^*_\cap|]\leq\mathbb{E}[\sum_{i\in[\mathsf{Cnt}^*],j\in[Q]}\mathbf{I}_{i,j}]=\mathsf{Cnt}^*\cdot\frac{\sqrt{n}}{10}\cdot\frac{1.1}{\sqrt{n}}$$

By Markov Inequality, the probability that $|T^*_\cap|\geq 0.5\mathsf{Cnt}^*$ is at most 0.5.

Summarizing the above, by the union bound, we have that

$$\Pr[\text{failure of finding a good match}]\leq 1/e^{20}+(1-1/e^{20})\cdot 0.5\leq 0.6$$

Therefore,

$$\begin{aligned}\Pr[Q\text{-th query is incorrect}]\leq&\Pr[\text{failure of }\mathsf{Resample}]+\Pr[\text{failure of finding a good match}]\\\leq&0.6+o(1)\leq 0.9\end{aligned}$$

Combining the above analysis of performence bounds, privacy and correctness, we get the following theorem.

Theorem 4.3. *There exists an information theoretic, perfectly private, and non-adaptively correct single-server preprocessing PIR scheme supporting $\Theta(\sqrt{n})$ bounded, random queries, and achieving the following performance bounds with all but $\mathsf{negl}(n)$ probability:*

- *there is an initial preprocessing phase in which the client makes a streaming pass over the database consuming with $O(n)$ bandwidth and server computation; further, the client performs $\widetilde{O}(n)$ computation;*
- *after the initial preprocessing, every query takes $\widetilde{O}(n^{1/3})$* online *bandwidth, $\widetilde{O}(n^{1/2})$* offline *bandwidth, $\widetilde{O}(n^{1/2})$ client computation, and $\widetilde{O}(n^{2/3})$ server computation; and*
- *the client space is $\widetilde{O}(n^{2/3})$.*

Further, in Appendix A we describe a new approach for upgrading the scheme to achieve *adaptive correctness* and moreover support *unbounded* and *arbitrary* queries, as stated in the following theorem.

Theorem 4.4 (Single-server preprocessing PIR for unbounded queries)**.** *There exists an information theoretic, perfectly private, and adaptively correct single-server preprocessing PIR scheme supporting unbounded, arbitrary queries, and achieving the following performance bounds with all but $\mathsf{negl}(n)$ probability:*

- *there is an initial preprocessing phase in which the client makes a streaming pass over the database consuming with $O(n)$ bandwidth and server computation; further, the client performs $\widetilde{O}(n)$ computation;*
- *after the initial preprocessing, every query takes $\widetilde{O}(n^{1/3})$* online *bandwidth, $\widetilde{O}(n^{1/2})$* offline *bandwidth, $\widetilde{O}(n^{1/2})$ client computation, and $\widetilde{O}(n^{2/3})$ server computation; and*
- *the client space is $\widetilde{O}(n^{2/3})$.*

5 Lower Bounds and Barriers

In this section we present two different limitations on single-server PIR with preprocessing. First, in Section 5.1, we show that schemes that achieve much better bandwidth/space tradeoff than the one given by Theorem 3.1 cannot exist information-theoretically, and in fact imply one-way functions (OWFs). This result holds even for significantly weaker notions of PIR with preprocessing as discussed below. For example, it also rules out certain forms of ORAM.

Then, in Section 5.2, we show a stronger implication for schemes that satisfy an additional "database obliviousness" requirement which is met by all known constructions. This implication gives a complexity theoretic barrier, showing that OWFs or even stronger "unstructured" hardness assumptions are unlikely to be sufficient for significantly improving the parameters of our Theorem 3.1 or the (OWF-based) Piano [ZPSZ24].

5.1 Single-Server Scheme with $o(\sqrt{n})$ Hint and Bandwidth Implies OWF

The General PIR Model and Result. We will make our lower bound stronger by considering a very weak notion of PIR with preprocessing and showing that it already implies OWFs.

Definition 1 ((n,t)-PIR)**.** For polynomials $n, t = \mathsf{poly}(\lambda)$ we define (n,t)-PIR as follows:

- There is a randomized PPT preprocessing procedure $(\widetilde{\mathbf{DB}}, \mathsf{hint}) \leftarrow \mathsf{Prep}(1^\lambda, \mathbf{DB})$ which is given the database $\mathbf{DB} \in \{0,1\}^n$. It generates a preprocessed database $\widetilde{\mathbf{DB}}$ for the server and hint for the client.

- There is a PIR protocol $\Pi(\widetilde{\mathbf{DB}}; (\mathsf{hint}, i_1, \ldots, i_t))$ between the PPT server who initially holds the preprocessed database $\widetilde{\mathbf{DB}}$ and the PPT client who initially has hint and t indices $i_1, \ldots, i_t$. At the end of the protocol the client should output $\mathbf{DB}[i_1], \ldots, \mathbf{DB}[i_t]$. The server and the client can maintain arbitrary local state during the protocol execution.

The *per-query communication complexity* c of Π is defined as C/t where C the total communication complexity of Π. We require two properties:

- **Correctness:** For any $\mathbf{DB} \in \{0,1\}^n$ and any $i_1, \ldots, i_t \in [n]$, if we sample $(\widetilde{\mathbf{DB}}, \mathsf{hint}) \leftarrow \mathsf{Prep}(1^\lambda, \mathbf{DB})$ and then run $\Pi(\widetilde{\mathbf{DB}}; (\mathsf{hint}, i_1, \ldots, i_t))$, then the client outputs the values $\mathbf{DB}[i_1]$, $\ldots, \mathbf{DB}[i_t]$ at the end of the protocol with overwhelming probability $1 - \mathsf{negl}(\lambda)$.

- **Security:** For any $\mathbf{DB} \in \{0,1\}^n$ and any two t-tuples of indices $(i_1^0, \ldots, i_t^0)$, $(i_1^1, \ldots, i_t^1) \in [n]^t$ chosen selectively, we have $\mathsf{view}_0 \approx_c \mathsf{view}_1$, where $\approx_c$ denotes computational indistinguishability and view_b is the protocol transcript of $\Pi(\widetilde{\mathbf{DB}}, (\mathsf{hint}, i_1^b, \ldots, i_t^b))$ for $(\widetilde{\mathbf{DB}}, \mathsf{hint}) \leftarrow \mathsf{Prep}(1^\lambda, \mathbf{DB})$.

Remarks. Note that the above notion of (n,t)-PIR relaxes PIR with preprocessing in many ways:

- There is a trusted preprocessing algorithm Prep that gives the server an encoded database $\widetilde{\mathbf{DB}}$, which may be arbitrarily correlated with the client's hint hint. This is useful in the contexts of ORAM [GO96] and private-key DEPIR schemes [CHR17, BIPW17]. In contrast, in our positive results and the previous related results from the literature [ZPSZ24, MIR23, GZS24a], as well as the definition from Section 2, preprocessing involves direct interaction between the client and the server, and the server only stores the original database $\mathbf{DB}$.

- Both the client and the server can be arbitrarily stateful during the protocol Π. Again, this even captures ORAM. Furthermore, the client storage can grow arbitrarily after the initial setup and our theorem only puts a bound on the size of the original hint given to the client during preprocessing.

- The number of queries is bounded by t rather than unlimited.

- The indices $i_1, \ldots, i_t$ are all chosen upfront and the protocol retrieves all of them together, rather than allowing the client to select them one by one.

- We only want a weak notion of security for an outsider who only sees the protocol transcript rather than the full view of the server itself. For example, the preprocessed database $\widetilde{\mathbf{DB}}$ is not included in the transcript, even though the server has it.

Our theorem says that in any PIR as above, we cannot simultaneously make the total communication of t queries be sublinear in n while having hint size sublinear in t, without one-way functions. In particular, for $t = \sqrt{n}$, this means that either communication complexity or the hint size must be $\Omega(\sqrt{n})$.

Theorem 5.1. *Any (n,t)-PIR with per-query communication complexity c and hint-size $|\mathsf{hint}| \leq t(1 - \frac{ct}{n}) - 1$ implies one-way functions. In particular, for $t = \sqrt{n}$, any information theoretic (n,t)-PIR must have either per-query communication complexity or hint size at least $\sqrt{n}/2$.*

For a preprocessing PIR scheme supporting unbounded queries, the above theorem also implies that *any consecutive window of t queries* must either have large total communication (at least $n/2$) or large client space (at least $t/2$).

To prove Theorem 5.1 we introduce a new intermediate notion of "mutual information amplification (MIA)" and show that PIR schemes imply MIA and that MIA implies one-way functions in certain parameter regimes.

Mutual Information Amplification. A mutual information amplification (MIA) protocol gives two correlated secrets $\mathsf{sk}_A, \mathsf{sk}_B$ to Alice and Bob respectively. Then Alice and Bob communicate over a public channel and their goal is to agree on a shared value k. In a non-trivial MIA, the shared k is computationally indistinguishable from one whose Shannon entropy conditioned on the view of the protocol is larger than the initial mutual information $I(\mathsf{sk}_A; \mathsf{sk}_B)$. We show that PIR schemes with good enough parameters imply non-trivial MIA and that non-trivial MIA implies OWFs.

Definition 2 (Mutual Information Amplification (MIA))**.** For polynomials $m, \ell = \mathsf{poly}(\lambda)$, a (m, ℓ)-MIA protocol consists of a PPT key generation algorithm $(\mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathbf{KGen}(1^\lambda)$ and a PPT protocol Π between Alice holding sk_A and Bob holding sk_B. At the end of the protocol Alice and Bob output values $\mathsf{k}_A, \mathsf{k}_B \in \{0,1\}^\ell$ respectively. We let $\Pi(\mathsf{sk}_A; \mathsf{sk}_B)$ denote the transcript of the protocol.

- **Correctness:** We require $\mathsf{k}_A = \mathsf{k}_B$ with overwhelming probability $1 - \mathsf{negl}(\lambda)$ over the randomness of $(\mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathbf{KGen}(1^\lambda)$ and the randomness of protocol execution Π.
- **Initial Mutual information:** $I(\mathsf{sk}_A; \mathsf{sk}_B) \le m$.
- **Amplification:** Given the protocol transcript, the shared key k_A is indistinguishable from having Shannon entropy ℓ. That is, there are some random variables $\mathsf{view}', \mathsf{k}'_A$ with

$$(\mathsf{view}, \mathsf{k}_A) \approx_c (\mathsf{view}', \mathsf{k}'_A)$$

  such that $H(\mathsf{view}') \ge H(\mathsf{view})$ and $H(\mathsf{k}'_A \mid \mathsf{view}') \ge \ell$ where $(\mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathbf{KGen}(1^\lambda)$, and $\mathsf{view}, \mathsf{k}_A$ correspond to the protocol transcript and Alice's output respectively in an execution of $\Pi(\mathsf{sk}_A; \mathsf{sk}_B)$.

In a non-trivial MIA, ℓ is noticeably larger than m.

PIR Implies Mutual Information Amplification. We show that PIR schemes with good enough parameters imply non-trivial MIA.

Theorem 5.2. *Any (n,t)-PIR scheme with per-query communication complexity c, gives a (m, ℓ)-MIA with $m \le |\mathsf{hint}|$ and $\ell \ge t(1 - \frac{ct}{n})$.*

Proof. Let $(\mathsf{Prep}, \Pi_{PIR})$ be an (n,t)-PIR and assume t divides n.[4] Define the MIA $(\mathbf{KGen}, \Pi_{MIA})$ as follows:

- $(\mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathbf{KGen}(1^\lambda)$: Choose $\mathbf{DB} \leftarrow \{0,1\}^n$ uniformly at random. Let $(\widetilde{\mathbf{DB}}, \mathsf{hint}) \leftarrow \mathsf{Prep}(1^\lambda, \mathbf{DB})$. Output $\mathsf{sk}_A = (\mathbf{DB}, \widetilde{\mathbf{DB}})$ and $\mathsf{sk}_B = \mathsf{hint}$.
- Π_{MIA}: A protocol where Alice holds sk_A and Bob holds sk_B.

[4] If not, we can always scale down n to some $n' \in [n-t, n]$ that is a multiple of t and do the proof for n'.

1. Bob samples $j \leftarrow \{0, \dots, n/t - 1\}$ and sends j to Alice. Define the indices $i_1 := j \cdot t + 1, \dots, i_t := (j+1)t$ to correspond to the j'th t-bit chunk of the database.
2. Alice and Bob run the protocol $\Pi_{PIR}(\widetilde{\mathbf{DB}}, (\mathsf{hint}, i_1, \dots, i_t))$ with Alice acting as the PIR server and Bob as the client. At the end of the protocol, Bob receives t bits $y_1, \dots, y_t$ and outputs $\mathsf{k}_B := (y_1, \dots, y_t)$
3. Alice outputs $\mathsf{k}_A := (\mathbf{DB}[i_1], \dots, \mathbf{DB}[i_t])$.

We will use the notation $(i_1, \dots, i_t) = \mathsf{chunk}(j)$ to denote the indices $i_\ell = j \cdot t + \ell$ and $\mathbf{DB}[\mathsf{chunk}(j)]$ to denote the database contents at those indices. The correctness of the above MIA follows from that of the PIR, which ensures $(y_1, \dots, y_t) = \mathbf{DB}[\mathsf{chunk}(j)]$ with overwhelming probability. The initial mutual information is

$$m = I(\mathsf{sk}_A; \mathsf{sk}_B) = I((\mathbf{DB}, \widetilde{\mathbf{DB}}); \mathsf{hint}) \leq |\mathsf{hint}| \tag{1}$$

as claimed. To argue amplification, we want to show that $(\mathsf{view}_{MIA}, \mathsf{k}_A) \approx_c (\mathsf{view}'_{MIA}, \mathsf{k}'_A)$ for some variables such that $H(\mathsf{view}'_{MIA}) \geq H(\mathsf{view}_{MIA})$ and $H(\mathsf{k}'_A | \mathsf{view}'_{MIA}) \geq \ell$. We have $\mathsf{view}_{MIA} = (j, \mathsf{view}_{PIR})$, where view_{PIR} is the transcript of $\Pi_{PIR}(\widetilde{\mathbf{DB}}, (\mathsf{hint}, \mathsf{chunk}(j)))$ and $\mathsf{k}_A := \mathbf{DB}[\mathsf{chunk}(j)]$. We define $\mathsf{k}'_A = \mathsf{k}_A = \mathbf{DB}[\mathsf{chunk}(j)]$ and $\mathsf{view}'_{MIA} = (j, \mathsf{view}'_{PIR})$ where view'_{PIR} is the transcript of a PIR protocol execution $\Pi_{PIR}(\widetilde{\mathbf{DB}}, (\mathsf{hint}, \mathsf{chunk}(j')))$ for a random and independent index $j' \leftarrow \{0, \dots, n/t - 1\}$. Computational indistinguishability follows directly from the security of PIR.

For the first entropy condition we have:

$$\begin{aligned} H(\mathsf{view}'_{MIA}) &= H(\mathsf{view}'_{PIR}) + H(j \mid \mathsf{view}'_{PIR}) \\ &= H(\mathsf{view}_{PIR}) + H(j) \\ &\geq H(\mathsf{view}_{PIR}) + H(j | \mathsf{view}_{PIR}) \\ &= H(\mathsf{view}_{MIA}), \end{aligned}$$

where the second line follows from the fact tat view'_{PIR} and view_{PIR} are identically distributed and j is independent of view'_{PIR}.

For the second entroy condition, first note that if the per-query communication complexity of the PIR is c, then we can bound the Shannon entropy:

$$\begin{aligned} H(\mathbf{DB} \mid \mathsf{view}'_{PIR}) &\geq H(\mathbf{DB}) - |\mathsf{view}'_{PIR}| \\ &\geq n - c \cdot t \\ H(\mathbf{DB} \mid \mathsf{view}'_{PIR}) &= \sum_{u \in \{0,\dots,n/t-1\}} H(\mathbf{DB}[\mathsf{chunk}(u)] \mid \mathbf{DB}[\mathsf{chunk}(0)], \dots, \mathbf{DB}[\mathsf{chunk}(u-1)], \mathsf{view}'_{PIR}) \\ &\leq \sum_{u \in \{0,\dots,n/t-1\}} H(\mathbf{DB}[\mathsf{chunk}(u)] \mid \mathsf{view}'_{PIR}) \end{aligned}$$

Therefore, for a random index j, we have:

$$\begin{aligned} H(\mathsf{k}_A \mid \mathsf{view}'_{MIA}) &= H(\mathbf{DB}[\mathsf{chunk}(j)] \mid (j, \mathsf{view}'_{PIR})) \\ &= \mathop{\mathbb{E}}_{u \leftarrow \{0,\dots,n/t-1\}} H(\mathbf{DB}[\mathsf{chunk}(u) \mid (j = u, \mathsf{view}'_{PIR})) \\ &= \mathop{\mathbb{E}}_{u \leftarrow \{0,\dots,n/t-1\}} H(\mathbf{DB}[\mathsf{chunk}(u)] \mid \mathsf{view}'_{PIR}) \\ &= (t/n) \sum_{u \in \{0,\dots,n/t-1\}} H(\mathbf{DB}[\mathsf{chunk}(u)] \mid \mathsf{view}'_{PIR}) \\ &\geq (t/n)(n - c \cdot t) \geq t(1 - \frac{ct}{n}) \end{aligned}$$

where the third line follows from the fact that $\mathbf{DB}, \mathsf{view}'_{PIR}$ are independent of j. □

Mutual Information Amplification implies One-Way Functions. We show that any non-trivial MIA implies one-way functions.

Theorem 5.3. *Any (m, ℓ)-MIA with $\ell \geq m + 1$ implies OWFs.*

Proof. Define the distribution $(\mathsf{view}, \mathsf{k}_A), (\mathsf{view}', \mathsf{k}'_A)$ as in the definition of security for MIA. Then:

$$\begin{aligned} H(\mathsf{k}_A|\mathsf{view}) &= H(\mathsf{k}_A) - I(\mathsf{k}_A; \mathsf{view}) && (2)\\ &\leq I(\mathsf{sk}_A; \mathsf{sk}_B) + H(\mathsf{k}_A|\mathsf{k}_B) && (3)\\ &\leq m + o(1) && (4)\end{aligned}$$

where (2) follows directly from the definition of mutual information, (3) follows directly from [Mau93, Theorem 1], and (4) follows from Fano's inequality: $H(\mathsf{k}_A|\mathsf{k}_B) \leq h(\varepsilon) + \varepsilon \cdot |\mathsf{k}_A| = o(1)$, where h is the binary entropy function and $\varepsilon = \Pr[\mathsf{k}_A \neq \mathsf{k}_B] = \mathsf{negl}(\lambda)$.

This implies that

$$\begin{aligned} H(\mathsf{view}', \mathsf{k}'_A) &= H(\mathsf{view}') + H(\mathsf{k}'_A|\mathsf{view}')\\ &\geq H(\mathsf{view}) + H(\mathsf{k}'_A|\mathsf{view}')\\ &\geq H(\mathsf{view}) + \ell\\ &\geq H(\mathsf{view}) + \ell + H(\mathsf{k}_A \mid \mathsf{view}) - m - o(1)\\ &\geq H(\mathsf{view}, \mathsf{k}') + 1 - o(1)\end{aligned}$$

So the distribution $(\mathsf{view}, \mathsf{k}_A)$ has noticable *false entropy* [IL89] meaning that it is computationally indistinguishable from a distribution $(\mathsf{view}', \mathsf{k}'_A)$ whose Shannon entropy is noticeably higher. By [IL89, Lemma 4.5], efficiently sampleable distributions with noticeable false entropy imply the existence of one-way functions. □

Proof of Theorem 5.1: An (n, t)-PIR with with per-query communication complexity c and hint-size $|\mathsf{hint}| \leq t(1 - \frac{ct}{n}) - 1$ implies (m, ℓ)-MIA with $m \leq |\mathsf{hint}| \leq t(1 - \frac{ct}{n}) - 1$ and $\ell \geq t(1 - \frac{ct}{n}) \geq m + 1$ by Theorem 5.2. Then, by Theorem 5.3 any such MIA implies OWFs. □

5.1.1 Extensions to the OWF Lower Bound

The above result in Theorem 5.1 can be further generalized along several dimensions.

Hint Size vs Mutual Information. Firstly, we notice that the condition that the hint size is small is only used in equation 1, and therefore we can replace it by the more general condition $I((\mathbf{DB}, \widetilde{\mathbf{DB}}); \mathsf{hint}) \leq t(1 - \frac{ct}{n}) - 1$, where $\mathbf{DB} \leftarrow \{0,1\}^n$ and $(\widetilde{\mathbf{DB}}, \mathsf{hint}) \leftarrow \mathsf{Prep}(1^\lambda, \mathbf{DB})$. For example, this means that our result also applies to schemes where $\widetilde{\mathbf{DB}} = \mathbf{DB}$ (the server just stores the original data) and the client hint includes an unlimited amount of randomness for free, as long as the database-dependent size of the hint is small. In other words, if the hint is of the form $\mathsf{hint} = (r, y = f(\mathbf{DB}, r))$ for some arbitrarily large randomness r and some small database-dependent part y then $I((\mathbf{DB}, \widetilde{\mathbf{DB}}); \mathsf{hint}) \leq |y|$ and therefore our lower applies to the database-dependent hint size $|y|$.

Index-Dependent Preprocessing. Secondly, the result even applies if the preprocessing $(\widetilde{\mathbf{DB}}, \mathsf{hint}) \leftarrow \mathsf{Prep}(1^\lambda, \mathbf{DB}, (i_1, \ldots, i_t))$ is allowed to depend on the indices $i_1, \ldots, i_t$ as well. The proof is identical.

Retrieving a Single Chunk. Thirdly, we only rely on a PIR scheme that can retrieve a single chunk of t consecutive locations in a database consisting of n/t such chunks. Therefore, we even get a lower bound on PIR schemes where the locations $i_1, \ldots, i_t$ must be consecutive locations of $\mathsf{chunk}(j)$ with $i_\ell = j \cdot t + \ell$. (Both correctness only needs to hold for such locations and security says we cannot distinguish PIR executions with locations $\mathsf{chunk}(j)$ vs. $\mathsf{chunk}(j')$ for arbitrary j, j').

Download vs. Communication. Fourth, the only place where we used that the per-query communication complexity is small was to bound

$$H(\mathbf{DB} \mid \mathsf{view}'_{PIR}) \geq H(\mathbf{DB}) - |\mathsf{view}'_{PIR}| \geq n - c \cdot t.$$

However, if we define d to be the per-query *download complexity* which only counts the communication from the server to the client, then we get

$$H(\mathbf{DB} \ \ |\mathsf{view}'_{PIR}) \geq n - d \cdot t - |\mathsf{hint}|.$$

This means that the theorem also applies when

$$|\mathsf{hint}| \leq t\left(1 - \frac{dt + |\mathsf{hint}|}{n}\right) - 1 \quad \Longleftrightarrow \quad |\mathsf{hint}|\left(1 - \frac{t}{n}\right) \leq t\left(1 - \frac{dt}{n}\right) - 1$$

When $t = \sqrt{n}$, this implies that either download complexity or the hint size must be $\Omega(\sqrt{n})$ in any information theoretic scheme.

5.2 SZK Barrier for Database-Oblivious Schemes

In this section we show that, for a natural class of schemes that captures all known constructions, the previous OWF implication can be significantly strengthened. Concretely, we give evidence that one-way functions or even stronger "unstructured" hardness assumptions are insufficient for significantly improving the bandwidth/space tradeoff of our Theorem 3.1 or Piano [ZPSZ24]. We start by defining the class of *database-oblivious* schemes to which our barrier applies. Note that all known constructions [ZPSZ24, MIR23, CK20, SACM21, LP22] are database-oblivious.

Definition 3 (Database-oblivious (n, t)-PIR)**.** A *database-oblivious* (n, t)-PIR scheme restricts Definition 1 in the following two ways:

- The database is not encoded, namely $\widetilde{\mathbf{DB}} = \mathbf{DB}$;
- Each question from the client to the server depends only on the client's randomness and the index sequence, but not on the server's responses.

Note that since we will only be concerned with hint size and bandwidth and not with server computation, eliminating the database encoding is not significant. In fact, Definition 3 only limits our main definition from Section 2 by restricting the client's questions to be independent of the server's responses (including the hint), which is a feature of our schemes as well as all relevant schemes from the literature.

Our barrier results hold even for a simpler restricted version of Definition 3 that we formalize next. This restriction only makes the following results stronger.

Definition 4 (Single-record (n, t)-PIR)**.** Let $n = n(\lambda), t = t(\lambda)$ be polynomially bounded functions such that $t|n$. A *single-record* (n, t)-PIR scheme is defined by polynomial-time algorithms (H, Q, A, D) with the following syntax:

- **Preprocessing:** The client, given security parameter λ, query index $i \in [n/t]$ and randomness $r \in \{0,1\}^\lambda$, obtains a hint $h = H(1^\lambda, \mathbf{DB}, i; r)$, where $\mathbf{DB}$ is a database consisting of n/t records of length t each.

- **Query:** The client computes a question $q = Q(1^\lambda, n, t, i; r)$; the server computes a response $a = A(\mathbf{DB}, q)$; the client outputs $d = D(a, h)$.

We make the following correctness and security requirements.

- **Correctness:** At end of the protocol, the client obtains the correct record $d = \mathbf{DB}[i]$ with $1 - \mathsf{negl}(\lambda)$ success probability.

- **Security:** For all sequences $i = i(\lambda), i' = i'(\lambda)$ with $i, i' \in [n/t]$, we have $Q(1^\lambda, n, t, i; r) \approx_c Q(1^\lambda, n, t, i'; r)$, where the probability is over the uniform choice of r.

Note that, assuming OWF exists, restricting r to be of length λ is without loss of generality. Indeed, the length of r can be extended using a pseudorandom generator without compromising the correctness and security requirements.

The above notion of single-record (n, t)-PIR can be viewed as a restricted instance of database-oblivious (n, t)-PIR from Definition 3. This follows from the fact that when $i_1, \dots, i_t$ are known in advance, database-obliviousness enables making all the questions to the server in a single round, independently of the database.

Claim 5.4. *Suppose there is a database-oblivious (n, t)-PIR with preprocessing download bandwidth α and query download bandwidth β. Then, there is a single-record (n, t)-PIR scheme with hint size $|h| = \alpha$ and query download bandwidth β.*

We now show that if the hint size is $o(t)$ and the query download bandwidth is $o(n)$, single-record (n, t)-PIR implies a simple variant of *somewhere statistical binding* (SSB) hash function [HW15, OPWW15], which we define next. Intuitively, such a hash function shrinks the input while being statistically binding on a chosen coordinate j, which is not revealed by the hash key.

Definition 5 (Vanilla SSB hash)**.** A *vanilla somewhere statistical binding (SSB) hash function* with (polynomially bounded) input length $n(\lambda)$ is defined by a pair of algorithms $(\mathsf{Gen}, \mathsf{Hash})$ with the following syntax:

- **Key generation:** Given a security parameter λ and index $j \in [n(\lambda)]$, the PPT algorithm $\mathsf{Gen}(1^\lambda, j)$ outputs a hash key hk, which we assume to contain λ.

- **Hashing:** Given a hash key hk and an input $x \in \{0,1\}^{n(\lambda)}$, the polynomial-time algorithm $\mathsf{Hash}(\mathsf{hk}, x)$ outputs a string y such that $|y| < |x|$.

We make the following binding and hiding requirements:

- **Weak local binding:** There is a computationally unbounded decoder Dec such that for every λ, $x \in \{0,1\}^{n(\lambda)}$, and $j \in [n(\lambda)]$ we have

$$\Pr[\mathsf{Dec}(\mathsf{hk}, j, \mathsf{Hash}(\mathsf{hk}, x)) = x_j \,:\, \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, j)] \geq 1 - \mathsf{negl}(\lambda).$$

- **Hiding:** For all sequences $j(\lambda), j'(\lambda) \in [n(\lambda)]$, we have $\mathsf{Gen}(1^\lambda, j) \approx_c \mathsf{Gen}(1^\lambda, j')$.

The above notion of vanilla SSB hash relaxes the previous notion from [HW15, OPWW15] in two ways: (1) it does not require an efficient local decommitment, (2) the usual binding requirement is weakened by assuming that hk is chosen independently of the input x. In fact, vanilla SSB hash is equivalent to a nontrivial standard 1-round single-server PIR (without preprocessing), where the client's decoding algorithm is computationally unbounded, the question size is not required to be sublinear, and there may be a negligible failure probability.

We now show how to construct a vanilla SSB hash function from a single-record (n, t)-PIR in the preprocessing model. The high-level idea is to replace the hint by applying a pairwise independent hash function to each database record.

Theorem 5.5 (From preprocessing PIR to vanilla SSB hash)**.** *Any single-record (n, t)-PIR scheme (H, Q, A, D) (Definition 4) with $t = \omega(\log \lambda)$, $|h| \le t/3$ and $|a| \le n/3$ implies a vanilla SSB hash (Definition 5), where $|h|$ and $|a|$ are the hint size and response size respectively. This holds even if D is computationally unbounded.*

Proof. Let $\mathcal{H} = \mathcal{H}(\lambda)$ be a family of universal hash functions $U : \{0,1\}^t \to \{0,1\}^{t/2}$. The SSB hash function is defined as follows.

- $\mathsf{Gen}(1^\lambda, j)$: Let $i = \lfloor j/t \rfloor$ (index of record containing the j-th database bit). Pick a uniformly random $r \in \{0,1\}^\lambda$ and let $q = Q(1^\lambda, n, t, i; r)$. Pick a random hash function $U \in_R \mathcal{H}$, and output $\mathsf{hk} = (q, U)$.
- $\mathsf{Hash}(\mathsf{hk} = (q, U), x)$: Parse the n-bit input x as a database with n/t records of length t. Output $y = (A(x, q), U(x[0]), \ldots, U(x[n/t - 1]))$.

First, note that the hash function is indeed shrinking: $|y| = |a| + (n/t) \cdot (t/2) \le n/3 + n/2 < n$. The hiding property of Gen is inherited from the security of Q.

To argue weak local binding, consider the following unbounded decoder $\mathsf{Dec}(\mathsf{hk}, j, y)$, where $\mathsf{hk} = (q, U)$ and $y = (a, y_0, \ldots, y_{n/t-1})$. First, compute the record index $i = \lfloor j/t \rfloor$. Then, for every possible hint h of length $|h| \le t/3$, compute a candidate $x_i^h = D(a, h)$ for $x[i]$. If there is a *unique* candidate x_i^h such that $U(x_i^h) = y_i$, output bit number $(j \bmod i)$ of x_i^h. Otherwise output $\perp$.

To see that the failure probability of the above Dec algorithm is negligible, first note that by the correctness of the PIR scheme, except with $\mathsf{negl}(\lambda)$ probability over the choice of r, there is a "correct" hint h for which $x_i^h = x[i]$ and hence $U(x_i^h) = y_i$. If this candidate for $x[i]$ is unique, Dec outputs the correct value of x_j. It remains to argue that the probability of an incorrect candidate $x_i^{h'} \neq x[i]$ satisfying $U(x_i^{h'}) = y_i$ is negligible. For each such incorrect candidate, the universality of U implies that $\Pr_{U \in \mathcal{H}}[U(x_i^{h'}) = U(x[i])] \le 2^{-t/2}$. Taking a union bound over all $\le 2^{t/3} + 1$ values of h', the probability of outputting $\perp$ is bounded by $(2^{t/3} + 1) \cdot 2^{-t/2} \le \mathsf{negl}(\lambda)$, as required. □

From vanilla SSB hash to a hard problem in SZK. Recall that our notion of vanilla SSB hash function can be viewed as a single-server PIR scheme (in the plain model) where: (1) the client's final decoding function is computationally unbounded and has a negligible failure probability, and (2) the download bandwidth is smaller than the database size n.

Liu and Vakuntanathan [LV16], in the context of arguing hardness of basing (standard) single-server PIR on NP $\not\subseteq$ BPP, showed how to use an SZK oracle to break the security of any nontrivial PIR scheme. Their proof does not rely on the PIR decoding algorithm being efficient, and is robust to a small decoding error (unlike, e.g., the PIR-based construction of collision-resistant hash functions [IKO05]). The SZK oracle they use is the (SZK-complete) entropy difference problem from [GV99], but their attack uses multiple oracle calls. Here we present a direct average-case hardness result by using a single oracle call to a different SZK problem from [Vad06].

Definition 6 (Conditional Entropy Approximation (CEA) problem)**.** The promise problem CEA is defined as follows. Given a circuit C, which samples a joint distribution (A, B), and an entropy threshold τ, decide whether

- Yes instances: $H(B \mid A) \geq \tau$
- No instances: $H(B \mid A) \leq \tau - 1$.

Lemma 5.6. *[Vad06] The promise problem* CEA *is in SZK.*

Theorem 5.7 (From vanilla SSB hash to hard SZK problem)**.** *Suppose a vanilla SSB hash function exists. Then there is a promise problem $\Pi(\lambda) = (X_Y, X_N)$ in SZK and $\mathsf{poly}(\lambda)$-time samplable distributions $\mathcal{X}_Y$, $\mathcal{X}_N$ that are supported by X_Y and X_N respectively (except with $\mathsf{negl}(\lambda)$ probability), such that $\mathcal{X}_Y \approx_c \mathcal{X}_N$.*

Proof. (sketch) We let $\Pi = \mathsf{CEA}$. The distributions $\mathcal{X}_Y, \mathcal{X}_N$ will be defined by $\mathsf{SSB} = (\mathsf{Gen}, \mathsf{Hash})$ as follows.

Let $n = n(\lambda)$ be the (polynomial) input length of SSB. Consider the experiment of picking uniformly random $x \in \{0,1\}^n$ and $j, j' \in [n]$, and letting $\mathsf{hk}_j = \mathsf{Gen}(1^\lambda, j)$ and $y = \mathsf{Hash}(\mathsf{hk}_j, x)$.

The weak binding property implies that $H(x_j \mid y) \leq \mathsf{negl}(\lambda)$. On the other hand, since Hash shrinks x, standard information theory inequalities (cf. [LV16]) imply that there is a polynomial p such that $H(x_{j'} \mid y) \geq 1/p(\lambda)$ (recalling that n is polynomial in λ).

For any fixed hash key hk and index $j \in [n]$, define a circuit $C_{\mathsf{hk},j}$ that samples a random $x \in \{0,1\}^n$ and outputs $(\mathsf{Hash}(\mathrm{hk}, x), x_j)$.

Define $\mathcal{X}_Y$ to output a circuit containing $\lambda \cdot p(\lambda)$ independent copies of $C_{\mathsf{hk}_j,j'}$ (with independent choices of j, j', hk_j in different copies) and $\mathcal{X}_N$ to output the same number of independent copies of $C_{\mathsf{hk}_j,j}$. In both cases, we set the entropy threshold to $\tau = 2$.

The hiding property of Gen implies that $\mathcal{X}_Y \approx_c \mathcal{X}_N$ as required. Finally, to argue that the support requirement is satisfied, note that the conditional entropy in the Yes case is distributed as the sum of $\lambda \cdot p(\lambda)$ independent instances of a random variable with values in $[0,1]$ and expectation $\geq 1/p(\lambda)$, which is ≥ 2 except with $\mathsf{negl}(\lambda)$ probability. Similarly, the conditional entropy in the No case is the sum of $\lambda \cdot p(\lambda)$ independent instances of a random variable with values in $[0,1]$ and expectation $\mathsf{negl}(\lambda)$, which is ≤ 1 except with $\mathsf{negl}(\lambda)$ probability. □

Combining Claim 5.4, Theorem 5.5 and Theorem 5.7, we have the following corollary.

Corollary 5.8 (SZK barrier for preprocessing PIR)**.** *Any database-oblivious (n,t)-PIR scheme (Definition 3) with $t = \omega(\log \lambda)$, client storage $t/3$ and total online download bandwidth $n/3$ implies a vanilla SSB hash function and an average-case hard promise problem in SZK. In particular, it implies a separation of (promise) SZK from BPP.*

Finally, we note that the above proofs fully relativize. Namely, if the database-oblivious (n,t)-PIR has access to an oracle $\mathcal{O}$, then it implies an SSB hash function with access to $\mathcal{O}$. The latter implies that $\mathsf{CEA}^{\mathcal{O}}$ (where the circuit C can make oracle calls to $\mathcal{O}$) separates (promise) $\mathrm{SZK}^{\mathcal{O}}$ from $\mathrm{BPP}^{\mathcal{O}}$. This gives a barrier even for *random oracle* constructions. Concretely, a ROM construction breaking the $O(\sqrt{n})$ barrier for database-oblivious (n,t)-PIR would imply a separation of SZK from BPP relative to a random oracle, settling a natural open problem in complexity theory.

Acknowledgements. We thank Aayush Jain, Ron Rothblum, Amit Sahai, Salil Vadhan and Prashant Nalini Vasudevan for discussions about SZK. We also would like to acknowledge helpful discussions with Ashrujit Ghoshal about PIR lower bounds. This research was initiated while the authors were visiting NTT Research. Yuval Ishai was supported by ERC grant NTSC (742754), BSF grant 2022370, ISF grant 2774/20, ISF-NSFC grant 3127/23, and a Google faculty research award. Daniel Wichs was supported by NSF grant CNS-1750795, CNS-2055510 and the JP Morgan Faculty Research Award. Elaine Shi was supported by NSF awards 2128519 and 2044679, an ONR grant, and a DARPA SIEVE grant under a subcontract from SRI.

References

[ABI86] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, dec 1986.

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *S&P*, 2018.

[Ajt10] Miklós Ajtai. Oblivious rams without cryptographic assumptions. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, STOC '10, pages 181–190, New York, NY, USA, 2010. ACM.

[BD19] Nir Bitansky and Akshay Degwekar. On the complexity of collision resistant hash functions: New and old black-box separations. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 422–450. Springer, 2019.

[BDRV18] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. From laconic zero-knowledge to public-key cryptography - extended abstract. In *Advances in Cryptology - CRYPTO*, volume 10993 of *Lecture Notes in Computer Science*, pages 674–697. Springer, 2018.

[BDV21] Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure versus hardness through the obfuscation lens. *SIAM J. Comput.*, 50(1):98–144, 2021.

[BFG03] Richard Beigel, Lance Fortnow, and William I. Gasarch. A nearly tight bound for private information retrieval protocols. *Electronic Colloquium on Computational Complexity (ECCC)*, 2003.

[BGIK22] Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I. Kolobov. Information-theoretic distributed point functions. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, volume 230 of *LIPIcs*, pages 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[BIM00] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: Pir with preprocessing. In *CRYPTO*, pages 55–73, 2000.

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In *TCC*, 2017.

[CG97] Benny Chor and Niv Gilboa. Computationally private information retrieval. In *STOC*, 1997.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *FOCS*, 1995.

[CHK22] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-server private information retrieval with sublinear amortized time. In *Eurocrypt*, 2022.

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In *TCC*, 2017.

[CK20] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In *EUROCRYPT*, 2020.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414, 1999.

[DG16] Zeev Dvir and Sivakanth Gopi. 2-server pir with subpolynomial communication. *J. ACM*, 63(4), 2016.

[DMN11] Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly secure oblivious RAM without random oracles. In *TCC*, pages 144–163, 2011.

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In *EUROCRYPT*, 2000.

[DPC23] Alex Davidson, Gonçalo Pestana, and Sofía Celi. Frodopir: Simple, scalable, single-server private information retrieval. *Proc. Priv. Enhancing Technol.*, 2023(1):365–383, 2023.

[Gas04] William I. Gasarch. A survey on private information retrieval. *Bulletin of the EATCS*, 82:72–107, 2004.

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 1996.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *ICALP*, 2005.

[GV99] Oded Goldreich and Salil P. Vadhan. Comparing entropies in statistical zero knowledge with applications to the structure of SZK. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, page 54. IEEE Computer Society, 1999.

[GZS24a] Ashrujit Ghoshal, Mingxun Zhou, and Elaine Shi. Efficient pre-processing pir without public-key cryptography. In *Eurocrypt*, 2024.

[GZS24b] Ashrujit Ghoshal, Mingxun Zhou, and Elaine Shi. Efficient pre-processing pir without public-key cryptography. In *Eurocrypt*, 2024.

[hav] https://haveibeenpwned.com/.

[HDCG$^+$23] Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, , and Nickolai Zeldovich. Private web search with Tiptoe. In *29th ACM Symposium on Operating Systems Principles (SOSP)*, Koblenz, Germany, October 2023.

[HHCG$^+$23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In *Usenix Security*, 2023.

[HMR12] Viet Tung Hoang, Ben Morris, and Phillip Rogaway. An enciphering scheme based on a card shuffle. In *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, page 1–13, Berlin, Heidelberg, 2012. Springer-Verlag.

[HW15] Pavel Hubácek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 163–172. ACM, 2015.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2005.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235. IEEE Computer Society, 1989.

[KO97] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *FOCS*, 1997.

[KY18] Ilan Komargodski and Eylon Yogev. On distributional collision resistant hashing. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO*, volume 10992 of *Lecture Notes in Computer Science*, pages 303–327. Springer, 2018.

[Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In *Information Security*, pages 314–328, 2005.

[Lip09] Helger Lipmaa. First CPIR protocol with data-dependent computation. In *ICISC*, 2009.

[LMRS23] Baiyu Li, Daniele Micciancio, Mariana Raykova, and Mark Schultz. Hintless single-server private information retrieval. *IACR Cryptol. ePrint Arch.*, page 1733, 2023.

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic ram computation from ring lwe. In *STOC*, 2023.

[LP22] Arthur Lazzaretti and Charalampos Papamanthou. Single server pir with sublinear amortized time and polylogarithmic bandwidth. Cryptology ePrint Archive, Paper 2022/830, 2022. https://eprint.iacr.org/2022/830.

[LV16] Tianren Liu and Vinod Vaikuntanathan. On basing private information retrieval on np-hardness. In *Theory of Cryptography - 13th International Conference, TCC*, volume 9562 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2016.

[Mau93] Ueli M. Maurer. Secret key agreement by public discussion from common information. *IEEE Trans. Inf. Theory*, 39(3):733–742, 1993.

[MCG$^+$08] Carlos Aguilar Melchor, Benoit Crespin, Philippe Gaborit, Vincent Jolivet, and Pierre Rousseau. High-speed private information retrieval computation on GPU. In *Proceedings of the 2008 Second International Conference on Emerging Security Information, Systems and Technologies*, SECURWARE '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[MG07] Carlos Aguilar Melchor and Philippe Gaborit. A lattice-based computationally-efficient private information retrieval protocol. *IACR Cryptology ePrint Archive*, 2007:446, 2007.

[MIR23] Muhammad Haris Mughees, Sun I, and Ling Ren. Simple and practical amortized sublinear private information retrieval. Cryptology ePrint Archive, Paper 2023/1072, 2023.

[MW22] Samir Jordan Menon and David J. Wu. SPIRAL: Fast, high-rate single-server PIR via FHE composition. In *IEEE S&P*, 2022.

[OG11] Femi G. Olumofin and Ian Goldberg. Revisiting the computational practicality of private information retrieval. In *Financial Cryptography*, pages 158–172, 2011.

[OPWW15] Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. New realizations of somewhere statistically binding hashing and positional accumulators. In *Advances in Cryptology - ASIACRYPT*, volume 9452 of *Lecture Notes in Computer Science*, pages 121–145. Springer, 2015.

[OS07] Rafail Ostrovsky and William E. Skeith, III. A survey of single-database private information retrieval: techniques and applications. In *PKC*, pages 393–411, 2007.

[OV08] Shien Jin Ong and Salil P. Vadhan. An equivalence between zero knowledge and commitments. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 482–500. Springer, 2008.

[PPY18] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Private stateful information retrieval. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 1002–1019. ACM, 2018.

[pri] Improving dns privacy with oblivious doh in 1.1.1.1. https://blog.cloudflare.com/oblivious-dns/.

[SACM21] Elaine Shi, Waqar Aqeel, Balakrishnan Chandrasekaran, and Bruce Maggs. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In *CRYPTO*, 2021.

[SC07] Radu Sion and Bogdan Carbunar. On the computational practicality of private information retrieval. In *Network and Distributed Systems Security Symposium (NDSS)*, 2007.

[SCSL11] Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with $O((\log N)^3)$ worst-case cost. In *ASIACRYPT*, 2011.

[sig] Technology deep dive: Building a faster oram layer for enclaves. https://signal.org/blog/building-faster-oram/.

[SvDS+13] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM – an extremely simple oblivious ram protocol. In *CCS*, 2013.

[Vad06] Salil P. Vadhan. An unconditional study of computational zero knowledge. *SIAM J. Comput.*, 36(4):1160–1214, 2006.

[WY05] David P. Woodruff and Sergey Yekhanin. A geometric approach to information-theoretic private information retrieval. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 275–284. IEEE Computer Society, 2005.

[Yeo23] Kevin Yeo. Lower bounds for (batch) pir with private preprocessing. In *Eurocrypt*, page 518–550. Springer-Verlag, 2023.

[ZLTS23] Mingxun Zhou, Wei-Kai Lin, Yiannis Tselekounis, and Elaine Shi. Optimal single-server private information retrieval. In *EUROCRYPT*, 2023.

[ZPSZ24] Mingxun Zhou, Andrew Park, Elaine Shi, and Wenting Zheng. Piano: Extremely simple, single-server pir with sublinear server computation. In *IEEE S& P*, 2024.

A Unbounded, Arbitrary Queries and Adaptive Correctness

The scheme in Section 4 supports only a window of $Q = \Theta(\sqrt{n})$ random and distinct queries. In this section, we describe an upgrade that allows us to support an unbounded number of arbitrary queries. We stress that unlike the earlier work Piano [ZPSZ24] that loses adaptive correctness during this upgrade. By contrast, our new upgrade technique preserves adaptive correctness, that is, the upgraded scheme still achieves correctness even when the queries are chosen adaptively by the adversary (i.e., server) based on its view so far. Note that privacy holds unconditionally even under adaptive queries, so we are mainly concerned about preserving correctness under adaptive queries in our upgrade.

Comparison with prior works' upgrade techniques. Instead of applying a separate pairwise-independent permutation to each hint entry, Piano [ZPSZ24] applies a single random permutation to the database upfront. For their approach to work, the query generation process must be independent of the choice of the global permutation. In other words, an adversary that adaptively chooses queries based on its view so far can easily break the correctness of their scheme. Mughees and Ren [MIR23] suggests an elegant idea that fixes the adaptive correctness issue of Piano [ZPSZ24]. Unfortunately, for technical reasons, their approach is not compatible with our new PIR construction.

A.1 Overview of Our Upgrade

The key stepping stone is to get a scheme that supports $Q = \Theta(\sqrt{n})$ *arbitrary* and distinct queries. If we can achieve this, we can easily upgrade it all the way to our final scheme which supports unbounded, arbitrary, and not necessarily distinct queries in the following way:

- *Removing the distinctness restriction.* further get rid of the distinctness requirement simply by having the client store the answers to the most recent Q queries, and if there is ever a duplicate query, the client simply looks up the answer locally, and issues a fake query that is distinct from all previous queries instead.

- *Supporting unbounded number of queries.* Now, to support an unbounded number of queries, we can simply rerun the preprocessing every Q queries. In fact, using a simple pipelining trick, we can deamortize the work of the next phase's preprocessing over the queries of the current phase.

A.2 Strawman Idea: Using Hint-Specific Permutations

To aid understanding, let us first consider the following inefficient idea:

- For each hint entry, we will associate it with a fresh random permutation $\pi : \{0, 1, \ldots, n-1\} \to \{0, 1, \ldots, n-1\}$. For the purpose of this hint entry alone, we use this permtuation to permute the database before mapping it to a (x, y, z) representation. In other words, given an index $id \in \{0, 1, \ldots, n-1\}$ into the database, it will now be mapped to the base-m representation of $\pi(id)$ where $m = n^{1/3}$ to produce the (x, y, z) coordinates.
- During a query, when the client sends (the succinct description of) the resampled set S' to the server, it also attaches the corresponding permutation π so the server will know how to map the databases indices to their corresponding (x, y, z) representations.

Since the permutation attached to each hint entry is not revealed to the server until the query time (i.e., after the query is selected), the choice of the permutation effectively randomizes the (x, y, z) representation of the query. Indeed, it is not hard to adapt our proof in Section 4.5 to work for this case as well.

Challenges. Unfortunately, there are two reasons why this idea causes significant performance penalties:

1. First, the description size of each random permutation π (without any hardness assumption) is linear. This means that the client would have to store a linear amount of information per hint entry and during each query, send a linear amount of information.
2. Second, this idea breaks the client's computational efficiency during preprocessing. The problem is that now, each database index $i \in \{0, 1, \ldots, n-1\}$ is known by each hint entry as a different (x, y, z) representation. As the client makes a streaming pass, we cannot even afford to compute the (x, y, z) representations of all indices for all hints — doing so would already result in $\Theta(\sqrt{n} \cdot n)$ computation, and amortizing over $O(\sqrt{n})$ queries would result in linear client computation per query!

It is not hard to see that $\Theta(\sqrt{n})$-wise independence is sufficient; however, using $\Theta(\sqrt{n})$-wise independence would still require linear client space, and moreover, it does not help solve the second problem mentioned above. With a little more work, we can show that in fact, an almost pairwise independent permutation family is sufficient for proving adaptive correctness. Since an almost pairwise independent permutation can be described in $\widetilde{O}(1)$ words [HMR12, ABI86]. we would be able to solve the first problem. Unfortunately, using a pairwise independent permutation still does not solve the second problem.

A.3 Our Idea: A Customized Random Permutation

To avoid the penalty in client computation during preprocessing, below we propose a special-purpose random permutation π which we attach to each hint. Our special-purpose random permutation is actually not pairwise independent, but it is sufficent for proving the key property we need, that is, Proposition 4.2. Further, the construction still admits efficient client preprocessing. Specifically, we can prove that the client's amortized work per query is only $O(n^{2/3})$ in the single-server setting.

Special-purpose random permutation π. Without loss of generality, we may assume that $n^{1/3} = 2^k$. Interpret an index $v \in [n]$ as $v = (x, y, z) \in \mathbb{GF}(2^k)^3$ for $k = \log n/3$. The random permutation $\pi(\cdot) := \pi_r(\cdot)$ is defined as follow: sample a random $r = (r_1, r_2, r_3) \in \mathbb{GF}(2^k)^3$ with $r_3 \neq 0$, and define:

$$\pi_r(x_\emptyset, y_\emptyset, z_\emptyset) = Rv \text{ where } R = \begin{bmatrix} r \\ r + (1,0,0) \\ r + (0,1,0) \end{bmatrix} \in \mathbb{GF}(2^k)^{3\times 3}.$$

In other words, $(x, y, z) = \pi_r(x_\emptyset, y_\emptyset, z_\emptyset)$ is defined by $x := r_1 \cdot x_\emptyset + r_2 \cdot y_\emptyset + r_3 \cdot z_\emptyset$, $y := x + x_\emptyset$, $z := x + y_\emptyset$. This is a permutation since R is *invertible* when $r_3 \neq 0$.

Claim A.1. *Fix arbitrary $v = (x_\emptyset, y_\emptyset, z_\emptyset)$, $v' = (x'_\emptyset, y'_\emptyset, z'_\emptyset)$ such that $v \neq v'$. Sample a random permutation π_r, let $(x, y, z) = \pi_r(x_\emptyset, y_\emptyset, z_\emptyset)$, and let $(x', y', z') = \pi_r(x'_\emptyset, y'_\emptyset, z'_\emptyset)$. Then, the following statements hold:*

$$\Pr_r[x = x'] \leq \frac{1}{n^{1/3} - 1}, \qquad \Pr_r[y = y'] \leq \frac{1}{n^{1/3} - 1}, \qquad \Pr[z = z'] \leq \frac{1}{n^{1/3} - 1}$$

Proof. Let $j \in \{1, 2, 3\}$ be an index for which $v_j - v'_j \neq 0$. Then, for the first part, we have

$$\Pr_r[x = x'] = \Pr_r[\langle r, v - v' \rangle = 0] \leq \Pr_{r_j}\left[r_j = (\sum_{i \neq j} r_i \cdot (v_i - v'_i))/(v_j - v'_j) \right] \leq \frac{1}{2^k - 1}.$$

The argument the y, z coordinates is the same with $r + (1, 0, 0)$ and $r + (0, 1, 0)$ taking the role of r and noting that their distribution is identical to that of r. □

Proposition A.2 (Counterpart of Proposition 4.2)**.** *Fix any original coordinates $(x_\emptyset, y_\emptyset, z_\emptyset)$ and $(x'_\emptyset, y'_\emptyset, z'_\emptyset)$ such that $(x_\emptyset, y_\emptyset, z_\emptyset) \neq (x'_\emptyset, y'_\emptyset, z'_\emptyset)$. For sufficiently large n,*

$$\Pr\left[\textit{Sample } (S, \pi) \textit{ s.t. } (x'_\emptyset, y'_\emptyset, z'_\emptyset) \in \mathsf{Set}(S, \pi) : \quad (x_\emptyset, y_\emptyset, z_\emptyset) \in \mathsf{Set}(S, \pi)\right] \leq 4.1/\sqrt{n}$$

In the above, $\mathsf{Set}(S, \pi)$ where $S = X \times Y \times Z$ is defined as follows: $(x_\emptyset, y_\emptyset, z_\emptyset) \in \mathsf{Set}(S, \pi)$ iff $\pi(x_\emptyset, y_\emptyset, z_\emptyset) \in X \times Y \times Z$.

Proof. Sample a random $(S = X \times Y \times Z, \pi)$ subject to $(x'_\emptyset, y'_\emptyset, z'_\emptyset) \in \mathsf{Set}(S, \pi)$. Let $(x, y, z) := \pi(x_\emptyset, y_\emptyset, z_\emptyset)$ and $(x', y', z') := \pi(x'_\emptyset, y'_\emptyset, z'_\emptyset)$. Henceforth, we use the notation $E(===)$ to denote the event $x' = x$, $y' = y$, and $z' = z$; similarly, we use the notation $E(=\neq\neq)$ to denote the event $x' = x$, $y' \neq y$, and $z' \neq z$, and so on.

$$\begin{aligned}
&\Pr[(x_\emptyset, y_\emptyset, z_\emptyset) \in \mathsf{Set}(S, \pi)] = \Pr[(x, y, z) \in X \times Y \times Z] && (5)\\
=&\Pr[x \in X, y \in Y, z \in Z | E(\neq\neq\neq)] \cdot \Pr[E(\neq\neq\neq)] && (6)\\
&+ \Pr[x \in X, y \in Y | E(\neq\neq=)] \cdot \Pr[E(\neq\neq=)] && (7)\\
&+ \Pr[x \in X, z \in Z | E(\neq=\neq)] \cdot \Pr[E(\neq=\neq)] && (8)\\
&+ \Pr[y \in Y, z \in Z | E(=\neq\neq)] \cdot \Pr[E(=\neq\neq)] && (9)\\
&+ \Pr[z \in Z | E(==\neq)] \cdot \Pr[E(==\neq)] && (10)\\
&+ \Pr[y \in Y | E(=\neq=)] \cdot \Pr[E(=\neq=)] && (11)\\
&+ \Pr[x \in X | E(\neq==)] \cdot \Pr[E(\neq==)] && (12)\\
&+ \Pr[E(===)] && (13)
\end{aligned}$$

In the above,

$$(6) \le \Pr[x \in X, y \in Y, z \in Z | E(\neq\neq\neq)] = (1/n^{1/6})^3 = 1/n^{1/2};$$

$$(7) = \Pr[x \in X, y \in Y | E(\neq\neq=)] \cdot \Pr[E(\neq\neq=)] \le (1/n^{1/6})^2 \cdot \frac{1}{n^{1/3} - 1} = o(1/n^{1/2})$$

Similarly, (8) $\le o(1/n^{1/2})$, and (9) $\le o(1/n^{1/2})$. Further,

$$(10) = \Pr[z \in Z | E(==\neq)] \cdot \Pr[E(==\neq)] \le \Pr[z \in Z | E(==\neq)] \cdot \Pr[x = x'] \le \frac{1}{n^{1/6}} \cdot \frac{1}{n^{1/3} - 1} \le 1.01/n^{1/2}$$

where the last inequality holds for sufficiently large n. Similarly, (11) $\le 1.01/n^{1/2}$, and (12) $\le 1.01/n^{1/2}$. Finally, (13) $= 0$. Summarizing the above, we get the desired proposition. □

A.4 Putting Everything Together

In our final scheme for unbounded, arbitrary queries, we run $\omega(\log n)$ independent instances of the following scheme in parallel, and for each query, the client can obtain the correct answer as long as there is one instance that gives the correct answer (which the client can detect).

The client runs the preprocessing algorithm upfront. Then, we proceed in windows of $Q = \sqrt{n}/10$ queries. In each window, the client piggybacks the preprocessing of the next window as it makes queries. The client caches the results of the current window of queries. If during the same window, it makes a duplicate query, it simply looks up the answer locally, and replaces the query with some arbitrary distinct query. Therefore, henceforth, we may assume that there are $Q = \sqrt{n}/10$ arbitrary, distinct queries. To support $Q = \sqrt{n}/10$ arbitrary, distinct queries, we make the following modification to the warmup scheme in Section 4:

- We now describe each hint as a pair (S, π) where S is a random set of expected size $n^{1/2}$ and π is a permutation that maps $\{0, 1, \ldots, n^{1/3} - 1\}^3$ onto itself, sampled from a special permutation family defined in Appendix A.3. Given an original database position expressed as $(x_\emptyset, y_\emptyset, z_\emptyset)$, we use $(x, y, z) := \pi(x_\emptyset, y_\emptyset, z_\emptyset)$ to determine the (x, y, z)-respresentation for the purpose of this hint. Recall that the set S has a succinct description of size $O(n^{1/6})$ and the permutation π can be described in $\widetilde{O}(1)$ bits.

- Whenever in the warmup scheme of Section 4, the client sends the succinct description of a resampled set S' to the server, it now sends both (the succinct description of) S' and the correponding permutation π. To compute the response, the server needs to call $(x_\emptyset, y_\emptyset, z_\emptyset) := \pi^{-1}(x, y, z)$ to map each hint-specific representation $(x, y, z) \in S'$ back to an original database index $(x_\emptyset, y_\emptyset, z_\emptyset)$. For the special case when the the client failed to find a matching entry from the hint table, it now sends a set S along with a permutation π both of which are sampled freshly at random.

With the hint-specific permutations, we need to modify the preprocessing algorithm as follows.

Efficient preprocessing algorithm. Using the permutation of Appendix A.3, we can consider the following preprocessing algorithm.

1. Proceed in batches. For each batch, download the next $n^{2/3}$ bits from the server denoted $\mathbf{DB}[x_\emptyset, *, *]$, and do the following:

   For each hint $X \times Y \times Z$, for each $x \in X$, if $y = x + x_\emptyset \in Y$: then for every $z \in Z$, let $(x_\emptyset, y_\emptyset, z_\emptyset) = \pi^{-1}(x, y, z)$, and update the parities of the planar sets $X \times \{y\} \times Z$, and $X \times Y \times \{z\}$ with $\mathbf{DB}[x_\emptyset, y_\emptyset, z_\emptyset]$.

2. Finally, for each hint $X \times Y \times Z$, compute the parities of $X \times Y \times Z$, $\{(X \backslash \{x\}) \times Y \times Z\}_{x \in X}$, $\{X \times (Y \backslash \{y\}) \times Z\}_{y \in Y}$, and $\{X \times Y \times (Z \backslash \{z\})\}_{z \in Z}$ from the parities of the planar sets.

Performance of preprocessing. Henceforth, we ignore the negligible probability that Fact 4.1 does not hold. Unless otherwise noted, the performance bounds hold with all but negligible probability.

For the above preprocessing algorithm, in each batch of Steps 1, for each hint, iterating over all $x \in X$ takes $O(n^{1/6})$ work, and the number of updates needed for each hint is the same as the number of indices in $[x_\emptyset, *, *]$ that are matched by the hint, which is in expectation $O(n^{2/3}/n^{1/2}) = O(n^{1/6})$. By taking a Chernoff bound over all $O(n^{1/2})$ hints, we get that for the entire batch, the client needs to perform $O(n^{1/2} \cdot n^{1/6}) = O(n^{2/3})$ work with all but negligible probability. Summing over all batches, Step 1 requires $O(n)$ total client work with all but negligible probability.

The work of Step 2 is only $O(n^{1/2} \cdot n^{1/6})$ and it is asymptotically dominated by the total work of Step 1. Summarizing, the total client work during preprocessing is $O(n)$ with all but negligible probability. It is also not hard to see that the client space is upper bounded by $O(n^{2/3})$.

Summarizing the above, we obtain the following theorem.

Theorem A.3 (Single-server preprocessing PIR for unbounded queries)**.** *There exists an information theoretic, perfectly private, and adaptively correct single-server preprocessing PIR scheme supporting unbounded, arbitrary queries, and achieving the following performance bounds with all but* $\mathsf{negl}(n)$ *probability:*

- *there is an initial preprocessing phase in which the client makes a streaming pass over the database consuming with* $O(n)$ *bandwidth and server computation; further, the client performs* $\widetilde{O}(n)$ *computation;*
- *after the initial preprocessing, every query takes* $\widetilde{O}(n^{1/3})$ online *bandwidth,* $\widetilde{O}(n^{1/2})$ offline *bandwidth,* $\widetilde{O}(n^{1/2})$ *client computation, and* $\widetilde{O}(n^{2/3})$ *server computation; and*
- *the client space is* $\widetilde{O}(n^{2/3})$*.*

Proof. For correctness and privacy, it suffices to show that the above scheme for bounded, arbitrary, distinct queries satisfies both privacy and correctness.

Privacy. The same privacy proof of Section 4.4 holds for this new scheme, despite each entry now having an extra permutation.

Adaptive correctness. The adaptive correctness proof follows in a similar fashion like the proof in Section 4.5. As before, we want to show that with probability at most 0.5, $|T^*_\cap| \geq 0.5|T^*|$.

The key step is to switch to a hybrid experiment that gets rid of the adaptiveness by using the privacy of the PIR scheme. Specifically, our privacy proof shows that even when conditioned on the adversary's view so far, T_{end} is distributed like a freshly sampled table of hints. Therefore, we can equivalently consider the following hybrid experiment which gets rid of the adaptiveness: the adversary first chooses the Q queries, then the challenger samples T_{end} at random. We want to show that in this hybrid experiment, with probability at most 0.5, $|T^*_\cap| \geq 0.5|T^*|$. The rest of the proof follows exactly like the proof in Section 4.5 except that Proposition 4.2 is now replaced with the new version Proposition A.2: Section 4.5 considers a *random* query that is distinct from the challenge query, whereas Proposition A.2 considers an *arbitrary* query distinct from the challenge query. Because the constant 4.1 in Proposition A.2 is slightly larger than the constant 1.1 in Proposition 4.2, we can conclude that with probability at most 0.6, $|T^*_\cap| \geq 0.7\mathsf{Cnt}^*$ which is sufficient to get 0.1 correctness probability.

Performance. The introduction of the hint-specific permutation π does not asymptotically increase the client space or bandwidth since π has a description size of $\widetilde{O}(1)$. For the preprocessing performance, the detailed performance analysis was presented earlier in Appendix A.4. The super-logarithmic factor in $\widetilde{O}(\cdot)$ comes from the $\omega(\log n)$-wise parallel repetition to boost the correctness probability to $1 - \mathsf{negl}(n)$. However, during the preprocessing phase, all $\omega(\log n)$ copies of the scheme can share the same streaming pass, so the preprocessing bandwidth and server work need not suffer from this super-logarithmic factor. $\square$

B Preprocessing PIR for the 2-Server Setting

Our techniques also directly imply an information theoretic 2-server preprocessing PIR scheme with $O(n^{2/3})$ client space and incurring $O(n^{1/2})$ client computation, $O(n^{2/3})$ server computation, and $O(n^{1/3})$ bandwidth per query. The main difference from the single-server scheme is that each preprocessing is now performed with a different server, which makes the preprocessing easier. As before, the client piggybacks the work of the next phase's preprocessing over this phase's queries, where each phase consists of $Q = \sqrt{n}/10$ queries. Henceforth, we use the following terminology: we assume that the client performs all queries with the *right* server, and performs preprocessing with the *left* server.

Specifically, for each preprocessing (piggybacked over queries except for the initial preprocessing), the client generates a hint table with L hints. It sends the hint table to the left server, which responds with all $O(n^{1/6})$ parities for the hint. The left server can compute all $O(n^{1/6})$ parities for a hint in $O(n^{1/2})$ time.

In this 2-server scheme, right-server privacy follows from the same proof of the single-server scheme — in both the single-server and 2-server schemes, the (right) server learns nothing during the preprocessing. Left-server privacy follows from the fact that its view consists of randomly sampled hints which can be simulated without knowing the client's queries. Correctness follows as in the single-server scheme.

This leads to the following theorem:

Theorem B.1 (2-server preprocessing PIR for unbounded queries)**.** *There exists an information theoretic, non-adaptively correct 2-server PIR scheme that supports an unbounded number of arbitrary queries, and achieving the following performance bounds:*

- *there is an initial preprocessing that requires $\widetilde{O}(n^{2/3})$ bandwidth, $\widetilde{O}(n^{2/3})$ client computation, and $\widetilde{O}(n)$ server computation;*
- *after the initial preprocessing, every query takes $\widetilde{O}(n^{1/3})$ online bandwidth, $\widetilde{O}(n^{1/6})$ offline bandwidth, $\widetilde{O}(n^{1/2})$ client computation, and $\widetilde{O}(n^{2/3})$ server computation;*
- *the client space is $\widetilde{O}(n^{2/3})$.*

Note that the 2-server scheme cannot benefit from the adaptive correctness techniques of Appendix A, because the preprocessing phase leaks the choice of the random sets to the left server. It is an interesting open question how to get a 2-server scheme with the above efficiency, but with adaptive correctness.