Lower Bounds for (Batch) PIR with Private Preprocessing

Kevin Yeo

Google and Columbia University, kwlyeo@google.com

Abstract. In this paper, we study *(batch) private information retrieval with private preprocessing*. Private information retrieval (PIR) is the problem where one or more servers hold a database of n bits and a client wishes to retrieve the i-th bit in the database from the server(s). In *PIR with private preprocessing* (also known as offline-online PIR), the client is able to compute a private r-bit hint in an offline stage that may be leveraged to perform retrievals accessing at most t entries. For privacy, the client wishes to hide index i from an adversary that has compromised some of the servers. In the *batch PIR* setting, the client performs queries to retrieve the contents of multiple entries simultaneously.

We present a tight characterization for the trade-offs between hint size r and number of accessed entries t during queries. For any PIR scheme that enables clients to perform batch retrievals of k entries, we prove a lower bound of $tr = \Omega(nk)$ when $r \geq k$. When $r < k$, we prove that $t = \Omega(n)$. Our lower bounds hold when the scheme errs with probability at most $1/15$ and against PPT adversaries that only compromise one out of ℓ servers for any $\ell = O(1)$. Our work also closes the multiplicative logarithmic gap for the single query setting ($k = 1$) as our lower bound matches known constructions. Our lower bounds hold in the model where each database entry is stored without modification but each entry may be replicated arbitrarily.

Finally, we show connections between PIR and the online matrix-vector (OMV) conjecture from fine-grained complexity. We present barriers for proving lower bounds for two-server PIR schemes in general computational models as they would immediately imply the OMV conjecture.

1 Introduction

Private information retrieval (also known as PIR) is a powerful cryptographic primitive that enables privacy-preserving retrieval of entries from a database held by one or more servers where a subset of the servers may be untrusted and colluding. For a database with n entries uniquely indexed by integers from $[n]$, PIR enables a client to retrieve the i-th entry of the database without revealing the query index i to the subset of colluding adversarial servers. The primitive of PIR

The full version of this paper may be found at [71].

was first introduced by Chor, Kushilevitz, Goldreich and Sudan [20] in the multi-server information-theoretic setting where the adversary compromises a strict subset of the servers with many follow-up work in this area (see [4,19,7,9,70,32,31] and references therein). Kushilevitz and Ostrovsky [51] first studied PIR in the single-server setting against computationally bounded adversaries. Many further works have also studied single-server PIR including [16,8,28,52,35,59,2,5,34,3,58] to list some examples.

PIR is an important cryptographic tool due to its endless implications to real-world settings. PIR has been used as a critical component in the design of many practical privacy-preserving applications such as advertising [38,66], communication [57,6], friend discovery [11], media consumption [40] and publish-subscribe systems [18] to list some examples.

Despite the potential applicability of PIR, the computational overhead of PIR remains a significant bottleneck that hinders wide spread usage of PIR in large-scale real-world settings. Beimel, Ishai and Malkin [10] proved that linear server computation is always required even in the multi-server setting where only a strict subset of servers is compromised.

In an attempt to surpass this barrier, many prior works have considered variants of PIR that have successfully overcome the linear server computation obstacle. We present two of these successful variants in PIR with preprocessing and batch PIR below.

PIR with Preprocessing. Beimel, Ishai and Malkin [10] introduced the notion of PIR with preprocessing where the server may compute a public r-bit hint in an offline, preprocessing stage. During query time, the server will aim to leverage the hint to answer PIR queries with sub-linear computational time t. We will denote this the *public preprocessing* setting as the hint is made available to the adversary's view. For this model, Beimel, Ishai and Malkin [10] presented constructions that had $O(n^{1/2+\epsilon})$ server time during queries but required polynomial $n^{O(1)}$ sized hints. On the other hand, Beimel, Ishai and Malkin [10] proved a $tr = \Omega(n)$ lower bound that was further improved to $tr = \Omega(n \log n)$ in [64]. This model has also been studied under the name of public-key doubly-efficient PIR [15]. There remains a large gap between the best upper and lower bounds for sub-linear server time $t = o(n)$ as the best upper bounds still require $tr = n^{O(1)}$.

As an analog, one can also consider the *private preprocessing* setting (also known as offline-online PIR) where the r-bit private client hint H is computed and stored by the client hidden from the adversary's view. This model has been studied in many works including [27,15,17,60,14,41,67]. Corrigan-Gibbs and Kogan [24] presented an upper bound of $tr = O(n)$. For example, this means that one can obtain sub-linear server time such as $t = \tilde{O}(\sqrt{n})$ using a $\tilde{O}(\sqrt{n})$-bit hint. The same work also proves a lower bound of $tr = \tilde{\Omega}(n)$. We note that there remains a multiplicative logarithmic gap between known constructions and lower bounds leading to the following question:

What is the optimal trade-off between hint size and server computation for PIR with private preprocessing?

Batch PIR. Another approach to obtain sub-linear server computation for PIR is to consider *batch queries*. In this setting, the client knows a batch of k entries that it wishes to retrieve ahead of time. The goal is to obtain amortized sub-linear server time across all k queries to beat the naive approach of executing k independent queries that results in $t = O(nk)$. Batch PIR has been studied in a large number of works including [10,47,62,39,43,56,6,5]. Excitingly, it has been shown that one can execute batch PIR queries with minimal overhead compared to single-query PIR. Ishai, Kushilevitz, Ostrovsky and Sahai [47] showed the existence of a batch PIR that uses total server time $t = \tilde{O}(n)$ when retrieving k entries simultaneously. Therefore, the amortized time per query is $\tilde{O}(n/k)$ that is sub-linear for sufficiently large k.

Combining Batching and Preprocessing. An intriguing idea to further improve the computational overhead of PIR would be to combine the techniques from batching and private preprocessing. First, we can take a look at what seems possible. As stated earlier, one can perform batch PIR queries with almost no overhead compared to single-query PIR. The dream would be to obtain the same result when performing batch queries for state-of-the-art PIR with private preprocessing schemes. In more detail, this dream construction would enable performing batch queries to k entries while maintaining the trade-off $tr = \tilde{O}(n)$ that results in amortized sub-linear query time $\tilde{O}(n/(rk))$ when using r-bit hints.

On the other hand, we can consider the efficiency achieved by straightforward approaches. The simplest construction is to execute k queries in parallel by storing k hints and performing k query algorithms resulting in $tr = \tilde{O}(nk^2)$. Another option is to perform k queries in sequence using a construction that enables multiple queries for a single preprocessing stage (such as the two-server schemes in [24,67]). This results in $tr = \tilde{O}(nk)$ but requires k rounds of client-server interaction. There remains a gap between the potential dream construction and the straightforward approaches. This leads to the following interesting question:

What is the optimal efficiency achievable by PIR schemes that utilize both batch queries and private prepocessing?

In this work, we address this question by providing a tight characterization of the trade-offs between the hint size and online server query time. We show that the dream construction is not possible and known approaches already achieve the optimal trade-off.

1.1 Our Contributions

In this paper, we will prove a tight characterization of the trade-offs between the hint size and the number of accessed (probed) entries during query time for PIR schemes that aim to combine batching and offline private preprocessing techniques. Note that any lower bound on number of probed entries is also a lower bound on server query time. We will present a lower bound that encompasses a wide range of constructions and matches the overhead of prior works.

Lower Bound. As our main result, we will prove a trade-off between the size of the private hint, r, computed in the offline preprocessing stage and the number of entries accessed (probed) by the server during online query execution, t, that also acts as a lower bound on the server time during queries. For a scheme that enables batches of k queries, we will show that $tr = \Omega(nk)$. To enable wider applicability of our result, we prove our lower bound for constructions with potentially multiple rounds of interaction during queries, non-zero error probabilities and/or inefficient preprocessing algorithms. Additionally, we consider weak PPT adversaries that only compromise one server. The following lower bounds are proven where the server(s) store the database is an unencoded manner, but may arbitrarily replicate entries (see Section 2.2 for more details).

Theorem 1 (Informal). *For any $\ell = O(1)$ and any k-query, ℓ-server batch PIR with private preprocessing scheme that errs with probability at most $1/15$ and is secure against a PPT adversary that compromises one server, it must be that $tr = \Omega(nk)$ when $k \leq r \leq n/400$. If $r < k$, it must be that $t = \Omega(n)$.*

The condition that $k \leq r \leq n/400$ is necessary to rule out trivial edge cases. There is a trivial setting where the entire database is stored in the hint using $r = n$. This would require $t = 0$ server time to retrieve any entries circumventing our lower bound. We avoid this edge case by enforcing that $r \leq n/400$. The choice of $n/400$ was for convenience and one may re-do our proofs to prove the same result for $r \leq n/c$ for constant $c \leq 400$. In the case that $r < k$, one can ignore the hint and execute a k-query batch PIR in $\tilde{O}(n)$ time matching our lower bound.

As our lower bound is for multiple servers, it immediately applies to the single server setting. Additionally, our result also applies to more powerful adversaries that compromise multiple (or all) servers or use infinite computational power.

We note that our lower bound immediately applies to the single-query setting where $k = 1$. As an immediate corollary, we get that:

Theorem 2 (Informal). *For any $\ell = O(1)$ and any ℓ-server, single-query PIR with private preprocessing scheme that errs with probability at most $1/15$ and is secure against a PPT adversary that compromises one server, it must be that $tr = \Omega(n)$ when $r \leq n/400$.*

This improves upon the previous known single query lower bounds [24] by multiplicative logarithmic factors closing the gap for single-query setting (for example, see Appendix A of [64] for a concrete matching instantiation). Our batch lower bound also improves upon the batch lower bounds from a concurrent work [22] by multiplicative logarithmic factors.

Finally, we note that our lower bounds may be directly applied to PIR with public preprocessing. The server may arbitrary store a r-bit hint encoding of the database. Then, our work proves a $tr = \Omega(n)$ lower bound. However, we note that better lower bounds of $tr = \Omega(n \log n)$ were proven in [64].

Upper Bound. In terms of constructions, we note that our lower bound has already shown that one of the straightforward approaches of performing k sequential queries is already optimal (up to logarithmic factors). However, this

construction requires k rounds of interaction between the client and the server. To obtain a single-round optimal construction, one can use the pipelined queries property of prior works [24,67,49] where each client query is independent of server responses and the client can issue multiple queries simultaneously. We refer readers to the full version for a more detailed description.

For completeness, we show that there also exists a blackbox reduction from single-query to batch PIR with private preprocessing without requiring the pipelined query property. Our reduction maintains a single round, but increases the overhead by additional logarithmic factors (unlike the pipelined query approach described above).

Theorem 3 (Informal). *Assuming the existence of a single-query, ℓ-server PIR with private preprocessing with $tr = f(n)$, there exists a k-query batch, ℓ-server PIR with private preprocessing scheme with a single-round query such that $tr = \tilde{O}(k \cdot f(n))$.*

Barriers to General Lower Bounds. Finally, we also study PIR lower bounds in general computational models without any restrictions on database storage. To do this, we present connections between PIR and the online matrix-vector OMV conjecture [45] from fine-grained complexity. We show a barrier to proving lower bounds for PIR schemes in general models. If one is able to prove a slightly weaker variant of our standard PIR model lower bounds in general models, it would immediately imply the online matrix-vector OMV conjecture.

Theorem 4 (Informal). *For any constant $\epsilon > 0$, suppose there exists no single-query, two-server PIR with private preprocessing with $tr = O(n^{1-\epsilon})$. Then, the online matrix-vector OMV conjecture is true.*

1.2 Technical Overview

Lower Bound. As our main result, we will lower bound the product of the private client hint size and the number of probed entries during query time. The core idea starts from considering any batch PIR with private preprocessing scheme that probes a sub-linear number of entries in the database. For simplicity, we will focus on the single-server setting with an information-theoretic adversary. Consider any scheme Π that only probes at most half the database (that is, $t \leq n/2$). For a database $D \in \{0,1\}^n$, consider a batch query $q \subseteq [n]$ to a random subset of k entries and the subset of entries that are probed when executing q, denoted by P. Note that the adversary sees the set of probed entries P. In the information-theoretic setting, it must be that the probed set P must be independent of the query q. As a result, we should expect that only half of the k random entries in q will also be probed (i.e., $|P \cap q| \leq k/2$). We show similar ideas still hold in the multi-server setting against PPT adversaries.

The above statement ends up providing a powerful way to compress the database. By probing and knowing the contents of at most t entries, the execution of query q will enable learning the contents of approximately $k/2$ entries for free.

In other words, we are able to design a compression algorithm for database D using Π whose compression performance directly relates to the query time of Π. As the query time t decreases, the rate of compression of our above algorithm increases. If we take D to be a uniformly random n-bit string, we can immediately get lower bounds on query time t and the hint size r as our algorithm should not be able to compress D beyond the information-theoretic minimum.

Finally, we note that there is a technical obstacle in designing the compression algorithm as described. The above description showed that one can compress using a single query to get the contents of $k/2$ entries for free. To get a strong compression rate, we need that each query recovers $\Theta(k)$ *new* entries for free. In other words, these discovered-for-free entries must not have been probed or queried by previous queries used by the compression algorithm. To overcome this obstacle, we show that picking uniformly random queries will enable discovery of $\Theta(k)$ new entries that were not previously probed or queried. As a result, it suffices for the compression algorithm to try a set of random queries to find the necessary query sequence that enables strong compression.

Comparison with [24] and [22]. We highlight the improvements in our proof techniques compared to the single-query lower bound in [24] and batch lower bound in [22] that enable logarithmically higher lower bounds. Our lower bound works directly with (batch) PIR whereas the prior works [24,22] uses abstractions through Yao's box problem. Therefore, we are able to prove stronger properties about (batch) PIR itself without worrying whether these properties also apply to the abstracted problem. Prior works [24,22] focus on Yao's box problem without utilizing any privacy properties of PIR. In particular, there is a mismatch between the requirements of Yao's box problem and the guarantees from PIR protocols. In the former, the querier is forbidden from directly probing boxes that are queried. For the latter, it is possible that all queried entries are probed directly but the probability must be small (see Lemma 3). Both prior works first prove lower bounds on Yao's box problem with some error probability and translate it to PIR later which results in losing logarithmic factors. In contrast, our work combines error probability directly with privacy properties of PIR enabling us to prove stronger lower bounds for (batch) PIR with private preprocessing.

Secondly, our paper proves stronger aggregate properties for multiple batch PIR queries. We show that there exists a set of multiple batch PIR queries such that at least k/5 probes performed by each query were not probed by any other batch PIR query in the set. As a result, we avoid unnecessarily encoding batch PIR queries that do not provide a significant number of free entries in our compression algorithms (see Step 5(b)ii in Figure 2). In contrast, [22] only proves properties of probed entries overlapping with queried entries for a single query in the multi-box extension of Yao's box problem (roughly corresponding to a single batch PIR query). Afterwards, the compression algorithm of [22] uses a greedy approach by adding the multi-box queries that probe the most new entries. Unlike our proof, this does not guarantee that queries with a small number of free entries are not encoded.

Upper Bound. For our single-round construction, we will leverage batch codes (introduced by Ishai, Kushilevitz, Ostrovsky and Sahai [47]) and any single-query PIR with private preprocessing scheme. At a high level, our construction uses batch codes to split up the database D into m buckets and perform a single-query PIR to each of the m buckets. We note this is a standard technique done in the past (see [47,6,5] as some examples). Batch codes guarantee that for any batch query $q = \{i_1, \ldots, i_k\} \subseteq [n]$, each of the i_j-th entries may be found in one of the m buckets. We will execute m parallel instances of single-query PIR with private preprocessing scheme for each of the m buckets where each instance uses (r/m)-bit hints. As the queries are done in parallel, we note our query algorithm uses a single-round of interaction. By plugging in a state-of-the-art batch code construction and single-query PIR with private preprocessing scheme, we obtain a single-round construction such that $tr = \tilde{O}(n)$ matching our lower bound.

Connection to Online Matrix-Vector Conjecture. Finally, we show barriers to proving lower bounds in general models of computation. To do this, we present a reduction from the online matrix-vector OMV conjecture [45] to PIR with private preprocessing. To do this, we start from the simple two-server, information-theoretic PIR with $O(\sqrt{n})$ communication [20] where the database is represented as a $\sqrt{n} \times \sqrt{n}$ matrix M. During query time, the client uploads vectors $a, b \in \{0,1\}^{\sqrt{n}}$ to each server respectively and the servers respond with matrix-vector multiplications Ma and Mb. We show that if there exists an efficient algorithm for solving OMV, each server can use the same algorithm to also provide answers for PIR with similar overhead. As a result, similar lower bounds that we have already proven for PIR with private preprocessing in general computational models would immediately imply that the OMV conjecture is true. In other words, this is a barrier as the OMV conjecture is a well-studied open problem and a core conjecture in fine-grained complexity.

1.3 Related Works

Private Information Retrieval. PIR is a heavily studied cryptographic primitive first introduced by Chor, Kushilevitz, Goldreich and Sudan [20] in the multi-server setting where it is assumed only a strict subset of servers are compromised and colluding. Many follow-up works have worked on improving the communication efficiency of multi-server PIR in the information-theoretic setting including [4,7,9,70,32]. The most communication-efficient scheme is by Dvir and Gopi [31] using matching vector codes [30]. Similar work has been done for computationally-secure multi-server PIR [19,36,12] where the most efficient constructions utilize function secret sharing techniques [13]. Single-server PIR was introduced by Kushilevitz and Ostrovsky [51] with many follow-up works including [16,8,28,52,35,59] aiming to improve efficiency or utilize different assumptions. Recent work has focused on optimizing the concrete efficiency of single-server schemes using lattice-based homomorphic encryption [2,5,34,3,58].

PIR with Preprocessing. PIR with public preprocessing was first introduced by Beimel, Ishai and Malkin [10] where the hint is public and studied in several follow-up works [15,64]. The PIR with private preprocessing model has

been studied in many works and under many different names including doubly-efficient PIR [15,17,14], private stateful information retrieval [60], private anonymous data access [41] and offline-online PIR [24,67]. For private preprocessing, Corrigan-Gibbs and Kogan [24] presented a construction with optimal trade-offs $tr = \tilde{O}(n)$ that was later extended to handle blocklist lookups efficiently [49]. Shi, Aqeel, Chandrasekaran and Maggs [67] presented logarithmic communication two-server schemes with optimal trade-offs. Further follow-up works have aimed to improve the efficiency in various dimensions [44,25].

Batch PIR. Several prior works have studied batch PIR to obtain efficient constructions using matrix multiplication [10,56], batch codes [47,62,42], the ϕ-hiding assumption [39] and list-decoding algorithms [43]. Recent works have considered practical constructions that utilize *probabilistic batch codes* [6,5] with error rates that are experimentally analyzed. These works obtain asymptotically optimal batch code parameters, but err on a subset of potential batch queries.

PIR Lower Bounds. Lower bounds for PIR have been studied for a variety of different complexity measures. Beimel, Ishai and Malkin [10] showed that server computation must be linear without any preprocessing even in the multi-server setting. Prior works have proven communication lower bounds for PIR [37,69].

In the public preprocessing setting, Beimel, Ishai and Malkin [10] showed that $tr = \Omega(n)$ that was improved to $tr = \Omega(n \log n)$ by Persiano and Yeo [64]. This is the highest lower bound possible for PIR with public preprocessing without implying higher lower bounds for branching programs (see [10,15]). For the private preprocessing model, Corrigan-Gibbs and Kogan [24] proved a lower bound $tr = \tilde{\Omega}(n)$ for a single query.

Compression Proofs. In our proof, we will use the incompressibility technique that has been used widely in the past. These were also used in prior PIR lower bounds [64]. To list some examples outside of PIR, incompressibility has been used in the studies of generic cryptographic constructions [33], one-way functions and PRGs [26], proofs of space [1], random oracles [68,29,21], the discrete logarithm problem [23] and oblivious data structures [54,48,63,46,61,50,65].

2 Definitions

2.1 Batch PIR with Private Preprocessing

We start by defining batch PIR with private preprocessing (also known as batch offline-online PIR). We present our formal definition:

Definition 1 (Batch PIR with Private Preprocessing). *A k-query batch PIR with private preprocessing scheme Π is a triplet of efficient algorithms $\Pi = (\Pi.\mathsf{Preprocess}, \Pi.\mathsf{Encode}, \Pi.\mathsf{Query})$ such that*

1. *$H \leftarrow \Pi.\mathsf{Preprocess}(R_H; D)$: The preprocessing algorithm is executed by the client and the server(s). The client receives the coin tosses R_H as input and the server(s) receives the database D as input to compute a preprocessed r-bit hint H. The hint H is privately stored by the client.*

2. $E \leftarrow \Pi.\mathsf{Encode}(D)$*: The encoding algorithm is executed by each server. Each server receives the database* D *and computes an encoding of the database* E*.*
3. $\mathsf{res} \leftarrow \Pi.\mathsf{Query}(q, H, R; E)$*: The query algorithm is jointly executed by the client and the server(s). The client receives as input the batch query of* k *entries* $q = \{i_1, \ldots, i_k\} \subseteq [n]$*, the hint* H *and coin tosses* R *while the server(s) receives the encoded database* E *as input. Once the query algorithm is complete, the client receives* res*, the algorithm's attempted response to query* q*.*

In the above definition, the query algorithm may be interactive and use multiple client-server roundtrips. We will prove our lower bound for query algorithms with unbounded round complexity to encompass more constructions. For our upper bounds, we will focus on single-round schemes for better efficiency.

Definition 2 (Standard PIR Model with Replications). *A* k*-query batch PIR with private preprocessing scheme* $\Pi = (\Pi.\mathsf{Preprocess}, \Pi.\mathsf{Encode}, \Pi.\mathsf{Query})$ *is in the standard PIR model if it satisfies Definition 1 and* $\Pi.\mathsf{Encode}$ *may replicate entries arbitraily and store a permutation of the replicated entries.*

Constructions in the above model ensure that the server stores each database entry without encoding. Additionally, the server can replicate entries arbitrarily and store a permutation of the copies thereafter. We will use the standard PIR model with replications for our lower bound (see Section 2.2 for more details).

Next, we will define the correctness of constructions. We define a query as correct if the query algorithm returns the correct contents for all k queried entries. If the contents of any of the k queried entries is incorrect, the answer is deemed incorrect. The error probability is defined as follows:

Definition 3 (Correctness). *A batch PIR with private preprocessing scheme* Π *errs with probability at most* ϵ *if, for every database* $D \in \{0,1\}^n$ *and query* $q \subseteq [n]$*, it holds that*

$$\Pr_{\mathbf{R}_H, \mathbf{R}}[\Pi.\mathsf{Query}(q, \mathbf{H}, \mathbf{R}; E) \neq (D_i)_{i \in q} \mid \mathbf{H}] \leq \epsilon$$

where $E \leftarrow \Pi.\mathsf{Encode}(D)$ *and* $\mathbf{H} \leftarrow \Pi.\mathsf{Preprocess}(\mathbf{R}_H; D)$*.*

We move on to formally defining the security of batch PIR with private preprocessing schemes. We consider adversaries $\mathcal{A}$ that compromise $1 \leq \ell_{\mathcal{A}} \leq \ell$ out of the ℓ total servers. When a server is compromised, the adversary $\mathcal{A}$ sees the request sent to the server as well as operations performed by the server. For the i-th server, we denote the adversary's view by transcript $\mathcal{T}_i$. We define security using the game in Figure 1. We denote $p^{\eta}_{\mathcal{A}}(D)$ as the probability that the adversary $\mathcal{A}$ outputs 1 in the game $\mathsf{IndGame}^{\eta}_{\mathcal{A}}(D)$ that is taken over the randomness of coin tosses R_H and R as well as any internal randomness of the adversary $\mathcal{A}$. Using this we define security as:

Definition 4 ($(\delta, \ell_{\mathcal{A}}, \ell)$-Security). *A* ℓ*-server batch PIR with private preprocessing scheme is computationally* $(\delta, \ell_{\mathcal{A}}, \ell)$*-secure if for all probabilistically polynomial time (PPT) adversaries* $\mathcal{A}$ *that compromise at most* $\ell_{\mathcal{A}}$ *servers and all sufficiently large databases* D*, the following holds:*

$$|p^0_{\mathcal{A}} - p^1_{\mathcal{A}}| \leq \delta(|D|).$$

$\mathsf{IndGame}^{\eta}_{\mathcal{A}}(D)$:

1. The challenger $\mathcal{C}$ runs $E \leftarrow \Pi.\mathsf{Encode}(D)$.
2. The adversary $(q^0, q^1, S) \leftarrow \mathcal{A}(D, E)$ on input the database D and encoded database E outputs two batch queries q^0 and q^1 as well as subset $S \subseteq [\ell]$ of the ℓ_A servers to compromise as the challenge.
3. The challenger $\mathcal{C}$ executes $H \leftarrow \Pi.\mathsf{Preprocess}(R_H; D)$ to obtain hint H using random coin tosses R_H and records transcripts $\mathcal{T}_1^p, \ldots, \mathcal{T}_\ell^p$.
4. The challenger $\mathcal{C}$ executes $\Pi.\mathsf{Query}(q^\eta, H, R; E)$ using random coin tosses R and records transcripts $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$.
5. The challenger $\mathcal{C}$ sends transcripts for all compromised servers, $\{\mathcal{T}_i^p, \mathcal{T}_i\}_{i \in S}$, to the adversary $\mathcal{A}$.
6. The adversary $\mathcal{A}(\{\mathcal{T}_i^p, \mathcal{T}_i\}_{i \in S})$ outputs a bit b.

Fig. 1. Security game for PIR with private preprocessing.

The above may be modified to consider statistical security by considering all computationally unbounded adversaries $\mathcal{A}$. We note that the *private* preprocessing is reflected by the fact that the adversary $\mathcal{A}$ does not receive the hint H as input and only the server's view of the interaction during the preprocessing phase. In contrast, the adversary would receive the hint H as input in the *public* preprocessing setting (see the definitions in [10,64]).

Finally, we define the efficiency of batch PIR with private preprocessing schemes. We will consider worst-case notions for all costs as follows:

Definition 5 ((r,t)-Efficiency). *A batch PIR with private preprocessing scheme is (r,t)-efficient if the following two properties hold:*

1. *For all databases $D \in \{0,1\}^n$ and coin tosses R_H, the hint H produced by $\Pi.\mathsf{Preprocess}(R_H; D)$ is at most r bits.*
2. *For all databases $D \in \{0,1\}^n$, queries $q \subseteq [n]$, random coin tosses R_H and R, the running time of $\Pi.\mathsf{Query}(q, H, R; E)$ is at most t where $H \leftarrow \Pi.\mathsf{Preprocess}(R_H; D)$ and $E \leftarrow \Pi.\mathsf{Encode}(D)$.*

2.2 Lower Bound Model

Standard PIR Model with Replications. The standard PIR model has been used to prove PIR lower bounds in prior works [10,24,64]. In the standard PIR model, the database is stored by the server(s) without any encoding. In our work, we will consider an extension where the server(s) may store an encoding of the database that consists of replicating various entries. We will refer to this extension as the *standard PIR model with replications*. For example, the database may permute the database a polynomial number of times and store a permutation thereafter. This covers replications of databases over multiple servers and the usage of systematic batch codes (see Section 4).

In our model, the client is able to store a r-bit hint that is computed in an preprocessing stage before the query. When measuring query time, the only cost is the number of entries that are probed or accessed. All other operations during query time can be performed free of charge including computation, accessing and generating randomness and accessing the hint. We only measure costs using server operations. So, our model enables clients to do all operations free of charge. We also note that our model does not account for the computational time needed to compute the hint. Therefore, our model applies to constructions even if their preprocessing algorithm is very computationally expensive. In terms of adversaries, we will only consider PPT adversaries that compromise one server. As we consider weak adversaries, our lower bound immediately implies to more powerful adversaries that may compromise multiple servers or use unbounded computational resources.

To our knowledge, the above model captures the most efficient constructions in many different categories. For example, the standard PIR model with replications is utilized by the most concretely efficient computational single-server PIR constructions using leveled FHE [2,5,3,58], two-server PIR constructions using function secret sharing [12,13], batch PIR [10,47,62,39,43,56,6,5] and all PIR with private preprocessing schemes [24,67,22]. Therefore, proving lower bounds in this model is important to understand the limitations of current techniques.

Barriers to General Models. We note that more expressive models are currently unable to prove high lower bounds for PIR. For example, we could consider the cell probe model that enables arbitrary encoding. Unfortunately, the highest lower bounds in the cell probe model peak at $\tilde{\Omega}(\log^2 n)$ [53] that are too low to prove meaningful lower bounds for PIR currently. We also note there are several barriers to proving similar lower bounds as ours with arbitrary server encoding. For example, such lower bounds would be one way to rule out the existence of some variants of program obfuscation. The constructions of Boyle, Ishai, Pass and Wootters [15] utilize server encoding and obfuscation to obtain sub-linear query time without any client storage and, thus, would beat our lower bound. In Section 5, we show that lower bounds in general models would also imply the online matrix-vector conjecture that is a core pillar of fine-grained complexity.

3 Lower Bound

In this section, we prove our lower bound for batch PIR with private preprocessing that characterizes the trade-offs between hint size and online query time for the server(s). We will prove the following theorem:

Theorem 5. *Let Π be a k-batch, ℓ-server PIR with private preprocessing scheme in the standard PIR model with replications that uses r-bit hints and probes at most t entries during online query time for any $\ell = O(1)$. Furthermore, suppose Π is computationally $(\delta, 1, \ell)$-secure for $\delta \leq 1/(25\ell)$ and Π errs with probability at most $\epsilon \leq 1/15$. If $k \leq r \leq n/400$, then $tr = \Omega(nk)$. Otherwise when $r < k$, then $t = \Omega(n)$.*

In our proof, we will assume that $k \leq r \leq n/400$. For the case when $r < k$, we will arbitrarily pad the hint until $r = k$. Even with this padding, note that the lower bound of $tr = \Omega(nk)$ immediately implies a lower bound of $t = \Omega(n)$. We note our lower bound applies for protocols with any number of round-trips. Additionally, we note that the choice of $\delta \leq 1/(25\ell)$ and $\epsilon \leq 1/15$ was for convenience. One can re-do our proofs for different constants. Our assumption of $\delta \leq (1/25\ell)$ being a constant is to improve our lower bound as it also applies to standard settings of negligible advantage for adversaries. Also, our results directly imply lower bounds against stronger adversaries that may compromise more than one server. Finally, we note that the assumption $r \leq n/400$ is necessary to rule out the trivial case where the entire database is stored in the hint and online queries do not need to probe any entries (i.e., $t = 0$). One can also re-do our proofs to also encompass larger choices of $r \leq n/c$ for smaller constant $1 < c < 400$.

To prove our main result, we will actually prove a variant of the theorem. Here, we will assume that the number of queries in each batch is larger than some constant and no larger than $n/10$. We formalize this in the following theorem:

Theorem 6. *Let Π be a k-batch, ℓ-server PIR with private preprocessing scheme in the standard PIR model with replications that uses r-bit hints and probes at most t entries during online query time where $\ell = O(1)$ and $k_c \leq k \leq n/10$ for some constant k_c. Furthermore, suppose Π is computationally $(\delta, 1, \ell)$-secure for $\delta \leq 1/(25\ell)$ and Π errs with probability at most $\epsilon \leq 1/15$. If $k \leq r \leq n/400$, then $tr = \Omega(nk)$. Otherwise when $r < k$, then $t = \Omega(n)$.*

It turns out this immediately implies our main theorem for any $k \geq 1$. In particular, we show that Theorem 6 immediately implies Theorem 5.

Proof of Theorem 5. To prove this, we show a reduction that any protocol Π for any $k \geq 1$ can be converted into a protocol Π' where $k_c \leq k' \leq n/10$ without any asymptotic overhead. Suppose there exists Π that beats Theorem 5. If $k_c \leq k \leq n/10$, we are already done.

If $k < k_c$, we can construct Π' for $k' = k_c$ from Π by executing $O(k_c/k)$ queries in parallel. This means storing $O(k_c/k)$ hints and running the query algorithm $O(k_c/k)$ times for each hint. As a result, $t' = t \cdot O(k_c/k)$ and $r' = r \cdot O(k_c/k)$. As $k \geq 1$ and $k_c = O(1)$, we know that $O(k_c/k) = O(1)$ and get that $t'r' = O(tr)$ with no additional asymptotic overhead to contradict Theorem 6.

When $n/10 < k \leq n$, we construct Π' for $k' = n/10$ from Π by arbitrarily padding queried entries that will be ignored. Then, execute a single query using Π. Note, this results in $t'r' = O(tr)$ with no additional asymptotic overhead as $k = \Theta(k')$ to contradict Theorem 6. □

Proof Overview. The rest of this section will be devoted to prove Theorem 6. Our proof of Theorem 6 will proceed in three steps:

1. First, we will characterize the relationship between queried and probed entries. Our goal is to show that not all queried entries can also be probed by leveraging the privacy requirements of PIR (Section 3.1).

2. Secondly, we show that random coin tosses and random batch queries allow for great compression by enabling to determine the contents of entries without ever requiring to probe the entry directly (Section 3.2).
3. Finally, we present an impossible compression scheme leveraging the above two facts that contradicts Shannon's source coding theorem to complete the proof (Section 3.3).

Additional Notation. For convenience, we will introduce additional notation that will be used throughout our proof. We will denote the set of entries probed by the set $\Pi.\mathsf{Probes}(q, D, H, R) \subseteq [n]$ for a batch query q, database D, hint H and coin tosses R. That is, $i \in \Pi.\mathsf{Probes}(q, D, H, R)$ if and only if the i-th entry of D is probed by at least one of the ℓ servers. Secondly, we will use $\mathbf{H}$ to represent a hint that is randomly generated by the preprocessing stage. That is, $\mathbf{H} \leftarrow \Pi.\mathsf{Preprocess}(D, \mathbf{R}_H)$ where $\mathbf{R}_H$ are uniformly random coin tosses. We will frequently write probabilities of the form $\Pr_{\mathbf{H},\mathbf{R}}[i \in \Pi.\mathsf{Probes}(q, D, \mathbf{H}, \mathbf{R})]$ that denotes whether the i-th entry is probed on any of the ℓ servers over the probabilities of randomly generated hints and coin tosses. In particular, this will be shorthand for the formal probability statement:

$$\Pr_{\mathbf{H},\mathbf{R}}[i \in \Pi.\mathsf{Probes}(q, D, \mathbf{H}, \mathbf{R})] = \Pr_{\mathbf{R}_H,\mathbf{R}}[i \in \Pi.\mathsf{Probes}(q, D, \mathbf{H}, \mathbf{R}) \mid \mathbf{H}]$$

where $\mathbf{H} \leftarrow \Pi.\mathsf{Preprocess}(\mathbf{R}_H; D)$. Additionally, we will use similar shorthand when analyzing error probabilities:

$$\Pr_{\mathbf{H},\mathbf{R}}[\Pi.\mathsf{Query}(q, \mathbf{H}, \mathbf{R}; D) \neq (D_i)_{i \in q}] = \Pr_{\mathbf{R}_H,\mathbf{R}}[\Pi.\mathsf{Query}(q, \mathbf{H}, \mathbf{R}; D) \neq (D_i)_{i \in q} \mid \mathbf{H}]$$

where $\mathbf{H} \leftarrow \Pi.\mathsf{Preprocess}(\mathbf{R}_H; D)$. In general, we will use $\mathbf{H}$ to represent a hint generated by providing random coin tosses $\mathbf{R}_H$ to $\Pi.\mathsf{Preprocess}(\mathbf{R}_H; D)$.

3.1 Characterizing Queried and Probed Entries

The main goal in this section is to characterize the set of probed entries and their relationship with the batch of k queried entries. At a high level, consider any Π that does not probe every entry in the database. For simplicity, suppose that Π probes only half the entries. Is it possible that the set of probed entries can be heavily correlated with the original set of k queries? For example, is it possible that Π can probe the entries corresponding to all k queries without being detected by an adversary? We resolve these questions here by providing a formal characterization between queried and probed entries.

To do this we start by making the following assumption:

$$t \leq n/(100\ell). \tag{1}$$

This is without loss of generality as otherwise our proof will already be complete as $t > n/(100\ell)$ immediately implies $tr = \Omega(nk)$ since $r \geq k$ and ℓ is constant.

Consider the t probed entries across all ℓ servers. For any of the n entries in the database, we will consider the i-th entry to be probed if the entry is probed

by at least one of the ℓ servers. Therefore, we know that at most $n/(100\ell)$ unique entries are probed for each online query. Intuitively, there should be a large fraction of the n entries that are probed only with $1/(100\ell)$ probability. We formalize this for a fixed k-batch query as follows:

Lemma 1. *Let $D \in \{0,1\}^n$ be any database. Fix k-batch query $\{1,\ldots,k\} \subseteq [n]$. Then, there exists a subset $S \subseteq [n]$ such that $|S| \geq n/2$ and for all $i \in S$, then*

$$\Pr_{\mathbf{H},\mathbf{R}}[i \in \Pi.\mathsf{Probes}(\{1,\ldots,k\}, D, \mathbf{H}, \mathbf{R})] \leq \frac{1}{50\ell}.$$

Proof. Towards a contradiction, suppose that this is false. That means, there exists strictly more than $n/2$ entries that are probed with probability at least $1/(50\ell)$. Then, we get that

$$t = \mathbb{E}_{\mathbf{H},\mathbf{R}}[|\Pi.\mathsf{Probes}(\{1,\ldots,k\}, D, \mathbf{H}, \mathbf{R})|] > \frac{n}{2} \cdot \frac{1}{50\ell} = \frac{n}{100\ell}.$$

This is a contradiction as we had assumed $t \leq n/(100\ell)$ in Equation 1. □

Next, we construct a polynomial time adversary $\mathcal{A}$ to distinguish queries to $\{1,\ldots,k\}$ and any other batch query. More formally, we construct a family of adversaries $\mathcal{A}_{i,q}$, for all $i \in [n]$ and $q \subseteq [n]$ below.

Adversary $\mathcal{A}_{i,q}$:

- **Challenge Phase** $\mathcal{A}_{i,q}(D)$:
  1. Return $(\{1,\ldots,k\}, q, \{x\})$ for uniformly random x from $[\ell]$.
- **Output Phase** $\mathcal{A}_{i,q}(\mathcal{T}_x^p, \mathcal{T}_x)$:
  1. Retrieve $\Pi.\mathsf{Probes}$ from $\mathcal{T}_x$ specifying all entries probed on the compromised server.
  2. Return 1 if and only if $i \in \Pi.\mathsf{Probes}$.

In other words, $\mathcal{A}_{i,q}$ is defined such that it compromises one of the ℓ servers uniformly at random, picks challenge queries q and $\{1,\ldots,k\}$ and returns 1 if and only if the i-th entry is probed. We prove the following lemma using $\mathcal{A}_{i,q}$.

Lemma 2. *Suppose that Π is computationally $(\delta, 1, \ell)$-secure. Fix any k-batch query $q \subseteq [n]$ and database $D \in \{0,1\}^n$. Let $S \subseteq [n]$ be as stated in Lemma 1 and suppose index $i \in S$. If $\delta \leq 1/(25\ell)$, then*

$$\Pr_{\mathbf{H},\mathbf{R}}[i \in \Pi.\mathsf{Probes}(q, D, \mathbf{H}, \mathbf{R})] \leq \frac{1}{25}.$$

Proof. Towards a contradiction, suppose that the statement is false. Consider adversary $\mathcal{A}_{i,q}$. For query $\{1,\ldots,k\}$, we know that $\mathcal{A}_{i,q}$ outputs 1 with probability at most $1/(50\ell)$ by Lemma 1 as $i \in S$. On the other hand, consider the probed entries on any query q. We know the probability that i is probed on at least one of the ℓ servers is strictly larger than $1/25$. As $\mathcal{A}_{i,q}$ compromises one server at

random, we know that $\mathcal{A}_{i,q}$ observes that the i-th entry is probed with probability strictly greater than $1/(25\ell)$. Therefore, $\mathcal{A}_{i,q}$ outputs 1 with probability strictly greater than $1/(25\ell)$. This contradicts the fact that any computational adversary has distinguishing advantage at most $\delta \leq 1/(25\ell)$ as the advantage is strictly greater than $1/(25\ell) - 1/(50\ell) = 1/(25\ell)$ to derive our contradiction. $\square$

Finally, we use the above lemma to prove our main characterization of probed entries in relation to the set of entries that are queried. In particular, we will prove an upper bound on the number of queried entries that are also probed.

Lemma 3. *Fix any database $D \in \{0,1\}^n$ and any k-batch query $q \subseteq [n]$. Let $S \subseteq [n]$ be as defined in Lemma 1. Then,*

$$\Pr_{\mathbf{H},\mathbf{R}}\left[\left|q \cap S \cap \Pi.\mathsf{Probes}(q, D, \mathbf{H}, \mathbf{R})\right| \leq \frac{|q \cap S|}{5}\right] \geq \frac{4}{5}.$$

Proof. Let $q = \{i_1, \ldots, i_k\}$ be the query to k indices $i_1, \ldots, i_k$. By Lemma 2,

$$\Pr_{\mathbf{H},\mathbf{R}}[i_j \in \Pi.\mathsf{Probes}(q, D, \mathbf{H}, \mathbf{R})] \leq \frac{1}{25}$$

whenever $i_j \in S$. Let $X_j = 1$ if and only if $i_j \in S \cap \Pi.\mathsf{Probes}(q, D, \mathbf{H}, \mathbf{R})$ and 0 otherwise. Note that $\mathbb{E}_{\mathbf{H},\mathbf{R}}[X_j \mid i_j \in S] \leq 1/25$. If we let X be the total number of queried entries of S that are also probed, that is $X = |q \cap S \cap \Pi.\mathsf{Probes}(q, D, \mathbf{H}, \mathbf{R})|$, then we know that $\mathbb{E}_{\mathbf{H},\mathbf{R}}[X] \leq |q \cap S|/25$ by linearity of expectation as $X = X_1 + \ldots + X_j$. By Markov's inequality, we get that

$$\begin{aligned}
&\Pr_{\mathbf{H},\mathbf{R}}\left[\left|q \cap S \cap \Pi.\mathsf{Probes}(q, D, \mathbf{H}, \mathbf{R})\right| \geq \frac{|q \cap S|}{5}\right] \\
&\quad = \Pr_{\mathbf{H},\mathbf{R}}\left[X \geq \frac{|q \cap S|}{5}\right] = \Pr_{\mathbf{H},\mathbf{R}}\left[X \geq 5 \cdot \frac{|q \cap S|}{25}\right] \leq \frac{1}{5}
\end{aligned}$$

since $\mathbb{E}_{\mathbf{H},\mathbf{R}}[X] \leq |q \cap S|/25$ to complete the proof. $\square$

The above lemma formally shows that at most (1/5)-fraction of queried entries that appear in S will also be probed with high probability.

Discussion about Model. For our lower bound, we design an adversary that must detect whether the i-th entry of the database is probed. This is possible in the standard PIR model with replications as the adversary knows the relationship between each probed entry and its original index in the database. If the database is encoded arbitrarily, this adversarial strategy is no longer possible. This is the main challenge in extending our result to general models.

3.2 Discovering Good Batch Queries

In this section, we will prove results about finding batch queries that will enable good compression of the database. At a high level, these *good* batch queries $q =$

$\{i_1, \ldots, i_k\} \subseteq [n]$ are ones that enable computing a large fraction of queried entries, D_{i_j}, without probing the i_j-th entry directly. That is, $i_j \notin \Pi.\mathsf{Probes}(q, D, H, R)$ for the used hint H and coin tosses R.

To formalize the above, we will aim to identify *good* sets for various entities. We start by defining the notion of goodness for sets of queries and randomness. We start by saying that a triplet of a k-batch query $q \subseteq [n]$, hint H and coin tosses R are *good* if and only if the following two properties hold:

1. (**Correctness.**) $\Pi.\mathsf{Query}(q, H, R; D) = (D_i)_{i \in q}$.
2. (**Discovery.**) $|q \cap \Pi.\mathsf{Probes}(q, D, H, R)| \leq 4k/5$.

We denote a triplet being good by $E^q(q, H, R) = 1$ if the above two conditions holds for the given triplet (q, H, R). Otherwise, we say $E^q(q, H, R) = 0$. Note, that the first property ensures correctness of retrieving the contents of all queried entries. The second property enables discovery. That is, at least $k/5$ of the queried entries are not probed directly during query execution.

Next, we move onto pairs of hints H and coin tosses R. In particular, we are interested in finding pairs (H, R) that enable the above properties for most queries. For this, we will focus on the query set Q that consists of all k-batch queries that aim to retrieve k distinct entries. We will denote $\mathbf{q}$ as the random variable that draws a query uniformly at random from Q. We say that pair of hint and coin toss H and R are good if and only if the following condition holds:

$$\Pr_{\mathbf{q}}[E^q(\mathbf{q}, H, R) = 1] \geq \frac{1}{30}.$$

We denote a pair H and R to be good by $E^R(H, R) = 1$ and $E^R(H, R) = 0$ for when it is not good. In other words, we say that a hint H and coin tosses R are good if and only if (1/30)-th of the queries $q \in Q$ exists such that q returns the correct answer and at most (4/5)-th of queried entries are probed when using the fixed hint H and coin toss R. In the next lemma, we show that a large fraction of pairs (H, R) are good and satisfy the above property.

Lemma 4. *Fix any database D. Then, we get that*

$$\Pr_{\mathbf{H},\mathbf{R}}[E^R(\mathbf{H}, \mathbf{R}) = 1] \geq \frac{1}{50}.$$

Proof. Thoughout the proof, we will denote $\mathbf{q}$ as being drawn uniformly from the query set Q. As Π errs with probability at most $\epsilon \leq 1/15$, we know that

$$\Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}[\Pi.\mathsf{Query}(\mathbf{q}, \mathbf{H}, \mathbf{R}; D) \neq (D_i)_{i \in \mathbf{q}}] = \epsilon \leq \frac{1}{15}.$$

By Lemma 3, we know that

$$\Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}\left[|\mathbf{q} \cap S \cap \Pi.\mathsf{Probes}(\mathbf{q}, D, \mathbf{H}, \mathbf{R})| \leq \frac{|\mathbf{q} \cap S|}{5}\right] \geq \frac{4}{5}.$$

We can bound the intersection size of q and S as follows. Let X be the number of indices in $q \setminus S$. As we know that $\mathbf{q}$ is chosen as a random k subset of $[n]$ and $|S| \geq n/2$, we get that $\mathbb{E}[X] \leq k/2$. By Markov's inequality, we know that $Pr[X \geq 3k/4] = \Pr[X \geq 3/2 \cdot k/2] \leq 2/3$. Therefore, we get that

$$\Pr_{\mathbf{q}}\left[|\mathbf{q} \cap S| \geq \frac{k}{4}\right] = 1 - \Pr_{\mathbf{q}}\left[X \geq \frac{3k}{4}\right] \geq \frac{1}{3}.$$

Then, we can see that if $|q \cap S| \geq k/4$ and $|q \cap S \cap \Pi.\mathsf{Probes}(q, D, H, R)| \leq |q \cap S|/5$, this immediately implies that $|q \cap \Pi.\mathsf{Probes}(q, D, H, R)| \leq |q \setminus S| + |q \cap S|/5 \leq 3k/4 + k/20 = 4k/5$. Therefore, we get that

$$\begin{aligned}
&\Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}\left[|\mathbf{q} \cap \Pi.\mathsf{Probes}(\mathbf{q}, D, \mathbf{H}, \mathbf{R})| \leq \frac{4k}{5}\right] \\
&\quad \geq \Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}\left[|q \cap S| \geq \frac{k}{4} \wedge |\mathbf{q} \cap \Pi.\mathsf{Probes}(\mathbf{q}, D, \mathbf{H}, \mathbf{R})| \leq \frac{|q \cap S|}{5}\right] \\
&\quad \geq \Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}\left[|q \cap S| \geq \frac{k}{4}\right] - \Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}\left[|\mathbf{q} \cap \Pi.\mathsf{Probes}(\mathbf{q}, D, \mathbf{H}, \mathbf{R})| > \frac{|q \cap S|}{5}\right] \\
&\quad = \frac{1}{3} - \frac{1}{5} = \frac{2}{15}.
\end{aligned}$$

Then, we get that

$$\begin{aligned}
&\Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}[E^q(\mathbf{q}, \mathbf{R}, \mathbf{H}) = 1] \\
&\quad \geq \Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}\left[|\mathbf{q} \cap \Pi.\mathsf{Probes}(\mathbf{q}, D, \mathbf{H}, \mathbf{R})| \leq \frac{4k}{5}\right] - \Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}[\Pi.\mathsf{Query}(\mathbf{q}, \mathbf{H}, \mathbf{R}; D) \neq (D_i)_{i \in \mathbf{q}}] \\
&\quad \geq \frac{1}{15}.
\end{aligned}$$

Towards a contradiction, suppose that $\Pr_{\mathbf{H},\mathbf{R}}[E(\mathbf{H}, \mathbf{R}) = 1] < 1/50$. This contradicts the prior inequality as we get that:

$$\begin{aligned}
&\Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}[E^q(\mathbf{q}, \mathbf{H}, \mathbf{R}) = 1] \\
&\quad \leq \sum_{x \in \{0,1\}} \Pr_{\mathbf{H},\mathbf{R}}[E^H(\mathbf{H}, \mathbf{R}) = x] \Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}[E^q(\mathbf{q}, \mathbf{H}, \mathbf{R}) = 1 \mid E^H(\mathbf{H}, \mathbf{R}) = x] \\
&\quad < \frac{1}{50} + \left(\frac{49}{50} \cdot \Pr_{\mathbf{q},\mathbf{H},\mathbf{R}}[E^q(\mathbf{q}, \mathbf{H}, \mathbf{R}) = 1 \mid E^H(\mathbf{H}, \mathbf{R}) = 0]\right) \\
&\quad < \frac{1}{50} + \frac{49}{50} \cdot \frac{1}{30} < \frac{1}{15}.
\end{aligned}$$

As a result, we know that $\Pr_{\mathbf{H},\mathbf{R}}[E(\mathbf{H}, \mathbf{R}) = 1] \geq 1/50$. □

Next, we will consider the intersection of a random query with an arbitrary subset X of at most $n/100$ entries. In particular, we expect that if we pick a random k-batch query from Q, then approximately $(1/100)$-th fraction of the

chosen queries would also appear in X. Later, we will use X to model previously probed and queried entries. In other words, we are aiming to show that a random query will not query too many entries that have been previously probed or queried. We formalize this in the following lemma:

Lemma 5. *For any subset $X \subset [n]$ such that $|X| \leq n/100$,*

$$\Pr_{\mathbf{q}}\left[|\mathbf{q} \cap X| > \frac{k}{10}\right] \leq \frac{1}{60}$$

where $\mathbf{q}$ is drawn uniformly at random from Q.

Proof. Note that we can model the choice of $\mathbf{q}$ as picking a random subset of size k from $[n]$. For any fixed $X \subset [n]$, we can see that

$$\begin{aligned}\Pr[|\mathbf{q} \cap X| > k/10] &\leq \frac{\binom{n/100}{k/10} \cdot \binom{n}{9k/10}}{\binom{n}{k}} \leq \frac{\left(\frac{en}{10k}\right)^{k/10} \cdot \left(\frac{10n}{9k}\right)^{9k/10}}{\left(\frac{n}{k}\right)^{k}} \\ &\leq \left(\frac{e}{10} \cdot \left(\frac{10}{9}\right)^{9}\right)^{k/10} \leq \frac{1}{60}\end{aligned}$$

where we use Stirling's approximation of binomial coefficients $(a/b)^b \leq \binom{a}{b} \leq (ea/b)^b$ and assuming that k is a sufficiently large constant □

3.3 An Impossible Encoding

Next, we use the characterization of probe probabilities from the prior two sections to construct an impossible compression for databases drawn from our hard distribution that we define below:

> **Hard Distribution.** Our hard distribution for databases, that we denote by $\mathbf{D}$, will be a uniformly random n-bit string. In other words, each of the n entries will be a uniformly random bit.

At a high level, our compression algorithm will leverage Π to efficiently recover as many queried entries without needing to probe the corresponding physical entry. We are able to do this by leveraging the formal characterization of Lemma 3 that shows that only a constant fraction of queried entries will also be probed with reasonably high probability. Then, we will leverage Lemmata 4 and 5 to find these *good* queries amongst random queries. In more detail, the compression algorithm will perform in rounds. In each round, the goal is to find a good k-batch query amongst a set of random k-batch queries such that a large portion of the contents of queried entries are unknown and will not be probed. Before going into more detail, we start by formalizing the model for presenting our compression scheme.

One-Way Communication Protocols. To formally prove our lower bound, we will consider a one-way communication protocol between parties Alice (the encoder) and Bob (the decoder). Alice will receive the input database. The goal of Alice is to send a single message to Bob that will enable Bob to always successfully decode the input database. Additionally, Alice and Bob will also receive the same shared randomness as input that will be used to help encode and decode the input database. In particular, the shared randomness will consist of the random coin tosses $\mathbf{R_{shared}}$ needed to execute Π and will be independent of the input database D.

Next, we prove a lemma that shows that the length of Alice's encoding cannot be much smaller than the minimum number of random bits stored in a database drawn from the hard distribution. We will consider prefix-free encodings where one possible message is not a prefix of another possible message.

Lemma 6. *For any one-way communication protocol where Alice's encodings are prefix-free and Bob is always able to decode the database D, it must be that*

$$\mathbb{E}[|\mathsf{Enc}(\mathbf{D})|] \geq n$$

where the randomness is over the hard distribution of databases $\mathbf{D}$, the shared random coin tosses $\mathbf{R_{shared}}$ and internal randomness of Alice's encoding algorithm.

Proof. To prove this lemma, we will utilize Shannon's source coding theorem that states the expected length of any prefix-free encoding scheme must be at least the entropy of the input conditioned on any shared inputs. In other words,

$$\mathbb{E}[|\mathsf{Enc}(\mathbf{D})|] \geq H(\mathbf{D} \mid \mathbf{R_{shared}}) = H(\mathbf{D}) = n.$$

The first equality is from the fact $\mathbf{R_{shared}}$ are random coin tosses independent of $\mathbf{D}$. The second equality uses that $\mathbf{D}$ is a uniformly random n-bit string. □

Discussion about Errors. In Lemma 6, we require that the encoding enables perfect decoding. However, this does not mean that we only prove lower bounds for schemes with zero error probability. In fact, we will utilize constructions that may err with probability as high $\epsilon \leq 1/15$ to build a perfect encoding scheme. To do this, our encoding will only rely on the PIR scheme for correct queries.

Encoding and Decoding Algorithms. We now formally present the encoding and decoding algorithms for our one-way communication protocol. Note that there are no computational bounds on the encoding and decoding algorithms. In particular, we will only care that Alice's encoding length is short in expectation and Bob is always able to decode the database.

First, we will formally describe the shared randomness $\mathbf{R_{shared}}$. In particular, $\mathbf{R_{shared}}$ will be broken into two parts. The first part will consist of random coin tosses $\mathbf{R}$ to execute a query. The second part will consist of uniformly random queries chosen from the query set Q. In particular, the second part will look of the form $(\mathbf{q}_1, \ldots, \mathbf{q}_s)$ where $s = 20n/(t + k/10)$. In other words, there will

Alice's Encoding: Alice receives database D and randomness $R_{\mathbf{shared}} = (R, q_1, \ldots, q_s)$ where $s = 20n/(t + k/10)$.

1. Set $H \leftarrow \Pi.\mathsf{Preprocess}(R_H; D)$ using random coin tosses R_H.
2. Set $A \leftarrow \emptyset$ to keep track of probed and queried entries.
3. Set P to be an empty string recording probed entries.
4. Set $S \leftarrow \emptyset$ to keep track of indices of successful query sets.
5. For $i = 1, \ldots, s$:
   (a) If $|S| \geq s/2000$:
      i. Terminate loop.
   (b) Check if q_i satisfies the following properties:
      i. (**Correctness.**) $\Pi.\mathsf{Query}(q_i, H, R; D) = (D_x)_{x \in q_i}$.
      ii. (**Overlap with Known Entries.**) $|q_i \cap A| \leq k/10$.
      iii. (**Overlap with Probes.**) $|q_i \cap \Pi.\mathsf{Probes}(q_i, D, H, R)| \leq 4k/5$.
   (c) If q_i does satisfies the above:
      i. Set $U \leftarrow \Pi.\mathsf{Probes}(q_i, D, H, R) \setminus A$ to be all probed entries that were not previously probed or queried in the order they were first probed.
      ii. If $|U| < t$, add entries in $[n] \setminus U$ to U in increasing index order until U contains t entries.
      iii. Set $S \leftarrow S \cup \{i\}$.
      iv. Set $P \leftarrow (P, (D_u)_{u \in U})$. That is, all entries in U in the order they were first probed followed by added entries in increasing order.
      v. Set $A \leftarrow A \cup U$.
      vi. Add the smallest $k/10$ indexed entries in $q_i \setminus A$ to A.
6. If $|S| < s/2000$, return the encoding $(0, D)$ as a 0-bit followed by a trivial n-bit encoding of D.
7. Set $\mathsf{Enc} \leftarrow (1, H, S, P)$ to be a 1-bit, the hint H using r bits and set S of successful query indices and P from the above loop. Note S requires $\log \binom{s}{s/2000}$ bits and P requires $ts/2$ bits to encode.
8. Set $\mathsf{Enc} \leftarrow (\mathsf{Enc}, \{D_x\}_{x \in [n] \setminus A})$. That is, the contents of all entries with indices in $[n] \setminus A$ in increasing index order. Note, each of the $s/2000$ successful queries adds exactly $t + k/10$ entries into A. So, the size of A at the end is $s/2000 \cdot (t + k/10) = n/100$ and this step requires $99n/100$ bits.
9. Return Enc.

Fig. 2. Description of Alice's encoding algorithm.

Bob's Decoding: Bob receives Alice's encoding and randomness $R_{\mathbf{shared}} = (R, q_1, \ldots, q_s)$ where $s = 20n/(t + k/10)$.

1. If the encoding starts with a 0-bit, decode D trivially and return D.
2. Decode H using the next r bits.
3. Decode $S \subseteq [s]$ using the next $\log \binom{s}{s/2000}$ bits.
4. Decode P using the next $st/2$ bits.
5. Set B to be the decoded database.
6. Set $A \leftarrow \emptyset$ to keep track of all indices that have been either probed or queried.
7. For $i \in S$ in increasing order:
   (a) Execute $\Pi.\mathsf{Query}(q_i, H, R; D)$. Even though Bob does not know the database entirely, Bob can complete this execution using Alice's encoding in the following way:
   (b) If Bob attempts to probe an entry $x \notin A$:
      i. Bob will use the next bit of P to decode D_x.
      ii. Set $B[x] \leftarrow D_x$.
      iii. Set $A \leftarrow A \cup \{x\}$.
   (c) If Bob attempts to probe an entry $x \in A$:
      i. Use $B[x]$ as the contents of the entry.
   (d) Set $B[x]$ to be the answer given by $\Pi.\mathsf{Query}(q_i, H, R; D)$ for $x \in q_i$.
   (e) If Bob probes less than $a < t$ entries outside of A, Bob uses the next $t - a$ bits in P to decode the smallest $t - a$ indexed entries outside of A. Then, Bob adds their contents into B and their indices in A.
   (f) Add the smallest $k/10$ indexed entries in $q_i \setminus A$ to A.
8. Decode $\{D_x\}_{x \in [n] \setminus A}$. Set $B[x] \leftarrow D_x$ for all $x \in [n] \setminus A$.
9. Return B.

Fig. 3. Description of Bob's decoding algorithm.

be s random batch queries from Q. Note that all of this shared randomness is independent of the database D.

Next, we describe Alice's encoding algorithm. The main goal of Alice is to compress the database D using Π. Alice will go through the random queries q_i with the goal of finding a query in the set that enables extracting entries in D without ever probing them. To do this, Alice aims to find queries that are correct with the caveat that the queried entries should not have been previously discovered and they will not be probed by the query itself when using random coin tosses R_H and R. When Alice finds such a *good* query that enables a high discovery rate, Alice will encode the identity of these queries and all necessary probed entries to let Bob simulate the queries as well. To do this, Alice will only encode contents of entries that were not previously discovered. Once Alice is able to find enough queries to encode at least $n/100$ of the entries in D, Alice will complete the encoding by sending the remaining undiscovered entries in D trivially. Alice's encoding algorithm is provided in Figure 2.

Next, we describe Bob's decoding algorithm. The goal of Bob is to simulate queries identically to Alice. To do this, Bob keeps track of all entries whose contents have been discovered. For each query encoded by Alice, Bob will aim

to execute the query without knowing the input database D. To do this, Bob performs probes one at a time. For any entries whose contents are known to Bob, Bob can simply use their contents and continue executing. For any probed entry that is unknown to Bob, Bob will use Alice's encoding to determine the contents. As we ensure Alice and Bob use the same hint H coin tosses R, it can be guaranteed that Alice and Bob execute queries identically. After executing all queries, Bob will simply decode all remaining unknown entries using the trivial encoding sent by Alice. Bob's decoding algorithm is provided in Figure 3.

Correctness. To see that the one-way communication protocol always enables Bob to decode the database, we will show that Alice and Bob simulate queries in the exact same way. In particular, Bob will only execute queries q_i that satisfy the conditions that are checked by Alice. For each of these queries, Bob will execute them identically to Alice as they use the same hint H and coin tosses R and any entries that are not known to Bob will be encoded by Alice. As a result, Alice and Bob will execute all of these queries identically. Furthermore, Alice and Bob will maintain identical sets A throughout the execution of their entire algorithms and Bob will be able to get the contents of all entries in A. Finally, as all entries outside of A are encoded trivially, we get that Bob is able to decode the database successfully.

Prefix-Free Encoding. Note any encoding starting with a 0-bit cannot be a prefix of an encoding starting with a 1-bit and vice versa. Therefore, it suffices to show the set of 0-bit encodings and 1-bit encodings are prefix-free independently. All encodings starting with a 0-bit are the same length of $1 + n$ and, thus, prefix-free. Similarly, all encodings prefixed with a 1-bit are the same length of $1 + r + \log \binom{s}{s/2000} + st/2000 + 99n/100$ bits and, also, prefix-free.

Encoding Length. Next, we analyze the length of Alice's encoding in bits. Our goal is to prove an upper bound on the expected encoding length. We break this down into two cases. The first case is when Alice's encoding starts with a 0-bit. In this case, Alice's encoding uses $1 + n$ bits. Next, we upper bound the probability that Alice's encoding starts with a 0-bit. This only occurs when $|S| < s/2000$. Consider any set of random queries $\mathbf{q}_1, \ldots, \mathbf{q}_s$. For each single query $\mathbf{q}_i$, we note that the probability that $\mathbf{q}_i$ satisfies the conditions of Step 5b of Alice's algorithm is at least

$$\Pr_{\mathbf{q}_i}[E^q(\mathbf{q}_i, H, R) = 1] - \Pr_{\mathbf{q}_i}[|\mathbf{q}_i \cap A| > k/10].$$

If we assume that the input $\mathbf{R}_H$ and $\mathbf{R}$ satisfy property that $E^R(\mathbf{H}, \mathbf{R}) = 1$ where $\mathbf{H} = \Pi.\mathsf{Preprocess}(D, \mathbf{R}_H)$, then we get that $\mathbf{q}_i$ satisfies the conditions of Step 5b of Alice's algorithm with probability at least

$$\Pr_{\mathbf{q}_i}[E^q(\mathbf{q}_i, H, R) = 1 \mid E^R(H, R) = 1] - \Pr_{\mathbf{q}_i}[|\mathbf{q}_i \cap A| > k/10] \geq \frac{1}{30} - \frac{1}{60} = \frac{1}{60}$$

by using Lemma 5 and the definition of $E^R(\mathbf{H}, \mathbf{R}) = 1$. Therefore, the probability that $\mathbf{q}_i$ satisfies the conditions of Step 5b is at least $1/60$. Then, we know that

$\mathbb{E}[|S|] \geq s/60$. Denote the probability of less than $s/2000$ queries succeeding by f, then

$$\begin{aligned} f &= \Pr\left[|S| < \frac{s}{2000} \mid E^R(H,R) = 1\right] \\ &= \Pr\left[|S| < \left(1 - \frac{97}{100}\right) \cdot \frac{s}{60} \mid E^R(H,R) = 1\right] \leq e^{-\frac{(97)^2 s}{(100)^2 60^2}} < \frac{4}{5} \end{aligned}$$

using Chernoff's bound, $s \geq 1000$ and the fact that the failure of each query set is an independent event. We get that $s \geq 1000$ by our assumptions that $t \leq n/(100\ell) \leq n/100$ as $\ell \geq 1$ and $k \leq n/10$ implying that $t + k/10 \leq n/50$. Therefore, Alice's encoding starts with a 0-bit with probability at most

$$\begin{aligned} &\Pr_{\mathbf{H},\mathbf{R}}[E^R(\mathbf{H},\mathbf{R}) = 0] + \Pr_{\mathbf{H},\mathbf{R}}[E^R(\mathbf{H},\mathbf{R}) = 1] \cdot \Pr[|S| < s/2000 \mid E^R(\mathbf{H},\mathbf{R}) = 1] \\ &\quad \leq \frac{49}{50} + \frac{1}{50} \cdot f < 1 \end{aligned}$$

where we use Lemma 4 to bound $\Pr_{\mathbf{H},\mathbf{R}}[E^R(\mathbf{H},\mathbf{R}) = 0] \leq 49/50$.

Next, we will analyze the case when Alice's encoding starts with a 1-bit. First, we show that it is always guaranteed that the condition $|A| = n/100$ will be true once Alice reaches the end of the encoding algorithm. Note that each successful query increases A by $t + k/10$ entries. Furthermore, Alice executes exactly $s/2000 = n/(100(t + k/10))$ queries. So, $|A| = s(t + k/10)/2 = n/100$.

All successful queries encode t probed entries. Encoding the indices of successful queries requires $\log \binom{s}{s/2000}$. Then, the encoding length is at most

$$\begin{aligned} 1 + r + \log \binom{s}{s/2000} + \frac{st}{2000} + (n - |A|) &\leq 1 + r + \frac{s \log(2000e)}{2000} + \frac{st}{2000} + (n - |A|) \\ &\leq 1 + r + \frac{n}{100} \cdot \frac{t + \log(2000e)}{t + k/10} + \frac{99n}{100} \end{aligned}$$

using the fact that $\binom{a}{b} \leq (ea/b)^b$. Then, we get that Alice's expected encoding length is at most

$$1 + fn + (1 - f) \cdot \left(r + \frac{n}{100} \cdot \frac{t + \log(2000e)}{t + k/10} + \frac{99n}{100}\right).$$

Completing the Proof. Finally, we complete the proof by combining the above analysis of Alice's expected encoding length combined with Lemma 6.

Proof of Theorem 6. By Lemma 6, we know that Alice's expected encoding length must be at least n bits. Therefore, we get the inequality

$$\begin{aligned} &1 + fn + (1 - f) \cdot \left(r + \frac{n}{100} \cdot \frac{t + \log(2000e)}{t + k/10} + \frac{99n}{100}\right) \geq n \\ \iff &(1 - f) \cdot \left(r + \frac{n}{100} \cdot \frac{t + \log(2000e)}{t + k/10} + \frac{99n}{100}\right) \geq (1 - f)n - 1 \end{aligned}$$

$$\iff r + \frac{n}{100} \cdot \frac{t + \log(2000e)}{t + k/10} \geq \frac{n}{100} - O(1)$$
$$\iff r(t + k/10) \geq \frac{nk}{2000} - O(t + k)$$
$$\iff rt \geq \frac{nk}{2000} - O(t + k) - \frac{rk}{10}$$

where we will assume that $k \geq 20 \log(2000e)$ and use the fact that $1/(1 - f) = O(1)$ as $f < 4/5$. We use that $r \leq n/400$ to see that $rk/10 \leq nk/4000$ and get

$$(r + \Theta(1))(t + \Theta(1)) \geq \frac{nk}{4000}$$

where we also use that $r \geq k \geq k_c$ for some sufficiently large but constant k_c. We assume that n is sufficiently large to get that $tr = \Omega(nk)$. □

4 Upper Bound

In this section, we present upper bounds for batch PIR with private preprocessing. As a reminder, we recall two ways to obtain optimal constructions matching our lower bound. First, one can execute k queries in sequence with any known single-query scheme [24,67] that uses k-rounds. Another option is to use pipelined queries approach utilizing a specific property of previous constructions [24,67,49] where client queries do not depend on server responses. In other words, clients may issue k queries simultaneously (see the full version for more details). In this section, we show that one can construct blackbox reductions from single-query to batch PIR without assuming any other properties of the single-query scheme.

4.1 Blackbox Single-Query to Batch Reduction

In the rest of this section, we will present constructions with single-round query algorithms. Our schemes will make blackbox usages of batch codes and single-query PIR with private preprocessing schemes. We note our described constructions make standard usage of batch codes for enabling batch PIR (see [47,6,5] for example) from any single-query PIR protocol. Our main adaptation is replacing the single-query PIR protocol with any single-query PIR with private preprocessing protocol.

Batch Codes. Batch codes are a primitive first introduced by Ishai, Kushilevitz, Ostrovsky and Sahai [47] that studies the problem of distributing a database of n bits into m buckets. The goal of the distribution is to enable any user to retrieve any batch of k entries by only querying at most t bits from each of the m buckets. Typically, the goal of batch codes is to minimize the total size of the encoding denoted by N.

There are several variants of batch codes that have been studied in the past. For our construction, we will focus on systematic batch codes that handle non-multiset queries. Systematic batch codes require that each symbol of the codeword to be an entry in the original database. Note, this ensures that our construction uses the same conditions as our lower model. Some batch codes handle

the more difficult setting of multiset queries where the same entry may be retrieved multiple times. For our protocol, we only require handling queries that retrieve k distinct entries. It is straightforward to handle batch PIR queries with duplicate entries using only non-multiset queries.

Additionally, we will only use systematic batch codes with $t = 1$. This means that only a single symbol in each bucket will be accessed. In other words, the decoding algorithm A is trivial as it will simply read one of the k desired entries from each of the m buckets. These codes have been referred to as replication-based batch codes [47] or combinatorial batch codes [62] in the past. Furthermore, we will assume that the decoding algorithm can obtain correct buckets and entries within the bucket without needing the database. This is a feature that is used for most usages of batch codes for batch PIR (such as [47,6,5]).

Definition 6 ([62]). *A $(n, N, k, m, 1)$ combinatorial batch code C is a set system $(X, \mathcal{B})$ where X is a set of n elements, $\mathcal{B} = (B_1, \ldots, B_m)$ is a collection of m subsets of X and a decoding algorithm C_A such that:*

1. *$N = \sum_{i \in [m]} |B_i|$. That is, the total length of all m subsets is at most N (where the length of each bucket is independent of x).*
2. *For each subset $\{x_1, \ldots, x_k\} \subseteq X$, $C_A(\{x_1, \ldots, x_k\}) = ((i_1, j_1), \ldots, (i_k, j_k))$ such that x_a is the j_a-the entry of B_{i_a} for all $a \in [k]$ and all of $(i_1, \ldots, i_k)$ are distinct.*

Protocol. We will formally present our single-round protocol in this section. At a high level, we will assume the existence of a $(n, N, k, m, 1)$ combinatorial batch code C that can handle non-multiset queries. Additionally, we will assume the existence of a single-query PIR with private preprocessing scheme that uses r-bit hints and t online query time. We will use both C and Π in a blackbox manner to construct our protocol.

Our protocol will first apply the batch code C to split up the database $D \in \{0,1\}^n$ into m buckets to get $C(D) = (B_1, \ldots, B_m)$ where each bucket contains a subset of entries from D, $B_j \subseteq D$, since C is systematic. By the guarantees of batch codes, we know that for every subset $\{i_1, \ldots, i_k\} \subseteq [n]$, there exists a subset $\{j_1, \ldots, j_k\} \subseteq [m]$ such that $D_{i_x} \in B_{j_x}$ for all $x \in [k]$. Next, instantiate m parallel instances of Π on each of the m buckets, $B_1, \ldots, B_m$, using hint lengths of r/m bits for all m instances. During query time for batch query $\{i_1, \ldots, i_k\} \subseteq [n]$, use the batch code to identify the subset $\{j_1, \ldots, j_k\} \subseteq [m]$ such that $D_{i_x} \in B_{j_x}$ for all $x \in [k]$. For each of these k buckets, perform a query to retrieve D_{i_x}. For the remaining $m - k$ buckets, retrieve any arbitrary entry. For the formal construction and proof, we refer readers to the full version.

Theorem 7. *Let C be a systematic $(n, N, k, m, 1)$ batch code and Π be a single-query PIR with private preprocessing scheme that is $(\delta, \ell_{\mathcal{A}}, \ell)$-secure with error probability ϵ and uses r-bit hints and online query time $t(n, r)$. Then, there exists a k-query batch PIR with private preprocessing scheme that is $(m\delta, \ell_{\mathcal{A}}, \ell)$-secure with error probability $k\epsilon$. If this construction uses r'-bit hints, then*

$$t'(n, r') = O(t(N_1, r'/m) + \ldots + t(N_m, r'/m))$$

where N_i is the number of bits in the i-th bucket of the encoding by C.

Relation to Lower Bound Model. Note that our lower bound model assumes that there is no encoding of the database beyond arbitrary replication. As systematic batch codes only require replication, the construction is compatible with our lower bound model.

Instantiations. Next, we pick different options for batch codes C and single-query PIR with private preprocessing Π to instantiate our construction. In this case, we do not require the feature that a single preprocessing stage handles multiple queries. As a result, we can use either any of the constructions in [24,67] to get the following constructions from batch codes.

Theorem 8. *Assuming the existence of a systematic $(n, N, k, m, 1)$ batch code and a single-query PIR with private preprocessing scheme that is $(\delta, 1, \ell)$-secure where δ is negligible and $\ell \in \{1, 2\}$ such that $tr = \tilde{O}(n)$ and is correct except with negligible probability, then there exists a single-round, k-query batch PIR with private preprocessing that is $(\delta, 1, \ell)$-secure where δ is negligible and $\ell \in \{1, 2\}$, uses r-bit hints and online query time t such that $tr = \tilde{O}(N \cdot m)$ and returns the correct answer except with negligible probability.*

For instantiating batch codes, many prior works (see [47,62,42,6,5] and references therein) have studied batch codes that may be used with PIR. For our purposes, we can use the $(n, O(n \log n), k, O(k), 1)$ batch code presented by Ishai, Kushilevitz, Ostrovsky and Sahai [47] that is built from unbalanced expanders. Using this batch code, we get the a construction with $tr = \tilde{O}(nk)$ that matches our lower bound (up to logarithmic factors).

Non-Explicit vs Explicit Batch Codes. As a caveat, the above used batch code is non-explicit. One can plug-in other explicit batch codes or explicit constructions of unbalanced expanders into the Ishai, Kushilevitz, Ostrovsky and Sahai [47] batch code to obtain an explicit batch PIR with private preprocessing. However, these explicit constructions will result in worse parameters.

5 Barriers for General Lower Bounds

In the prior sections, we prove all our lower bounds in the standard PIR model as done in prior works (such as [10,24,64]). Recall that in the standard PIR model, the database of n entries must be stored without any modification. A more ambitious goal would be to prove lower bounds without restrictions on the underlying PIR algorithms. For example, a natural goal is to extend the above lower bounds to the case where the PIR algorithm may encode and store the database of n entries arbitrarily but using only a bounded amount of preprocessing time such as polynomial $n^{O(1)}$ time. We denote this the *general PIR model* where the PIR construction may store the database of n entries in any encoded manner. During query time, we denote the online time as the total amount of online computation performed by the server to process client queries. See the full version for more details on the general PIR model.

We will study connections between PIR and the online matrix-vector OMV conjecture from fine-grained complexity. We show barriers for proving lower bounds in general models as we show that such lower bounds would immediately imply the OMV conjecture that is a well-studied open problem in the theoretical computer science community and a core pillar of fine-grained complexity.

5.1 Online Matrix-Vector OMV Conjecture

At a high level, the OMV problem receives a single matrix $n \times n$ matrix M as input that may be preprocessed in polynomial time. Afterwards, n vectors $v_1, \ldots, v_n$ will be given in an online fashion such that the multiplication fo Mv_i must be output before receiving the next vector in the stream. We formally define the online matrix-vector problem below:

Definition 7 (Online Matrix-Vector Multiplication). *The online matrix-vector multiplication problem* OMV *takes as input a matrix $M \in \{0,1\}^{n \times n}$ and a stream of vectors $v_1, \ldots, v_n \in \{0,1\}^n$. The goal is to output Mv_i over $\mathbb{F}_2$ before seeing any input vectors $v_{i+1}, \ldots, v_n$.*

As matrix-vector multiplication is a fundamental problem in algorithms, the problem has been well-studied. A trivial algorithm would be to perform no preprocessing and simply answer each Mv_i naively requiring $O(n^3)$ total time. To date, the best known algorithm requires total time $n^3/2^{\Omega(\sqrt{\log n})}$ by Larsen and Williams [55]. However, this algorithm specifically requires using properties of the Boolean semiring. To our knowledge, there remains no better algorithm for OMV over any field including $\mathbb{F}_2$. Due to lack of progress, it has been conjectured [45] that there is no algorithm for solving the OMV problem in truly sub-cubic $O(n^{3-\epsilon})$ time. We formally present the OMV conjecture below:

Definition 8 (Online Matrix-Vector Conjecture [45]). *For any constant $\epsilon > 0$, there does not exist any algorithm with total online query time $O(n^{3-\epsilon})$ that can solve the online matrix-vector multiplication problem with error probability at most $1/3$ even with $n^{O(1)}$ preprocessing time of the matrix M.*

The online matrix-vector OMV conjecture is a very important conjecture in the area of fine-grained complexity that aim to quantify hardness within P along with other important conjectures such as the strong exponential time hypothesis (SETH), the 3SUM conjecture and the all-pairs shortest path (APSP) problem. The OMV conjecture has been used to prove the tight conditional lower bounds for several dynamic data structure problems including reachability and single-source shortest paths in directed graphs [45].

Boolean Semiring vs. $\mathbb{F}_2$. The standard definition of OMV is defined over the Boolean semiring where addition and multiplication are replaced by Boolean OR and AND operations (see [45,55] for example). In our work, we focus on the matrix-vector multiplication over $\mathbb{F}_2$. However, we note that the Boolean semiring variant of OMV is not easier than the $\mathbb{F}_2$ version of OMV. In the full version, we present a reduction showing that any algorithm solving OMV over $\mathbb{F}_2$ may be used to solve OMV over the Boolean semiring for completeness.

5.2 Barriers for General PIR Lower Bounds

We start by showing that we can construct efficient PIR constructions using efficient algorithms for online matrix-vector multiplication OMV. The result of this reduction is a barrier to proving lower bounds for PIR in more general bounds. In particular, we show that if one were able to prove lower bounds of $tr = \Omega(n)$ from Section 3 without the restrictions of the standard PIR model, this would immediately imply that the online matrix-vector conjecture is true.

We present a reduction from OMV that enables constructing efficient two-server PIR with public preprocessing schemes. Note that we had only focused on private preprocessing schemes. The main difference is that public preprocessing enables the adversary to view the execution and output of the preprocessing algorithm. We highlight that any lower bound for PIR with private preprocessing immediately implies the identical lower bound for public preprocessing. See the full version for formal definitions.

Theorem 9. *If an online matrix-vector multiplication algorithm running in total time $t(n)$ with $p(n)$ preprocessing time and error probability at most $\epsilon(n)$, then there exists a single-round, perfectly-secure, two-server PIR with public preprocessing with $O(p(n))$ preprocessing time, $O(t(\sqrt{n})/\sqrt{n})$ amortized online time over $\sqrt{n}$ queries, $O(\sqrt{n})$ communication and error probability at most $\epsilon(n)$.*

Proof. For this reduction, we start from a slight adaptation of the $O(\sqrt{n})$ communication two-server PIR scheme [20]. We arrange the database of n entries into a $\sqrt{n} \times \sqrt{n}$ matrix M. Suppose that we wish to query for an entry (i, j) in the i-th row and j-th column of M. The client will generate a uniformly random vector $a \in \{0,1\}^{\sqrt{n}}$ where each entry is either 0 or 1 with probability 1/2. Next, the client will compute $b = a \oplus 1_j$ that flips the bit in the j-th entry of a. The vector a is sent to the first server and the vector b is sent to the second server. Afterwards, each server will compute the matrix-vector multiplication $a' = Ma$ and $b' = Mb$ over $\mathbb{F}_2$. Note that $a'[i] \oplus b'[i] = M[i][j]$ and, thus, the client can retrieve the desired entry. For privacy, note that each vector a and b are identical to uniformly random vectors meaning the scheme is perfectly-secure.

Now, suppose that we have an algorithm $\mathcal{A}$ for the OMV problem with total time $t(n)$ over n queries and preprocessing time $p(n)$. We instantiate $\mathcal{A}$ with the $\sqrt{n} \times \sqrt{n}$ matrix M representing the n entries of the database in each of the two servers requiring $2p(n)$ preprocessing time. When responding to queries, each server will utilize $\mathcal{A}$ to compute the matrix-vector multiplication. Therefore, the total time to answer $\sqrt{n}$ queries is $2 \cdot t(\sqrt{n})$ and, thus, the amortized time is $O(t(\sqrt{n})/\sqrt{n})$. Note, the client is unable to retrieve the desired entry only when at least one of the responses from the server is not correct. By parallel repetition and taking the majority answer, we can drive down the error probability to be at most $\epsilon(n)$ without affecting the overall overhead except by constant factors. Finally, the query algorithm requires a single round completing the proof. □

Next, we present the corollary of the above theorem showing that lower bounds for PIR with private preprocessing would immediately prove that the OMV conjecture is true. See the full version for the proof.

Corollary 1. *For any constant $\epsilon > 0$, suppose there does not exist any two-server computationally-secure PIR with private preprocessing scheme in the general model that uses $n^{O(1)}$ preprocessing time, r-bit hints, amortized online time t over $\sqrt{n}$ queries and error probability at most $1/3$ such that $tr = O(n^{1-\epsilon})$. Then, the online matrix-vector* OMV *conjecture is true.*

We note that the above was presented to focus on PIR with private preprocessing. One can generalize the above theorem in several ways. First, the above corollary still holds even if we only considered PIR with public preprocessing time with truly sub-linear $t = O(n^{1-\epsilon})$ time. Secondly, the result still holds even if the lower bound applied only to perfectly-secure two-server PIR schemes. Finally, the result also holds for any k-server scheme where $k \geq 2$.

Discussion about Public Preprocessing. Prior works have also proven lower bounds for public preprocessing including [10,64]. All these lower bounds are also proven in the standard PIR model. The results in this section also show that proving lower bounds for PIR with public preprocessing in more general models would immediately imply that the OMV conjecture is true.

6 Conclusions and Open Problems

In this paper, we present a tight characterization of the trade-offs between the hint size and online query time for batch PIR with private preprocessing. In particular, we present a $tr = \Omega(nk)$ lower bound when retrieving k entries. On the other hand, we show the existence of a $tr = \tilde{O}(nk)$ single-round query construction. In other words, our results show that one can only reap the benefits of the techniques from one of batch PIR or PIR with private preprocessing. When ignoring private preprocessing (i.e. $r < k$), we can apply known batch PIR techniques and get that $t = \tilde{\Theta}(n)$. For optimal PIR with private preprocessing schemes with $tr = \tilde{\Theta}(n)$ and $r \geq k$, one cannot beat the efficiency of the naive approach of performing k queries sequentially to get $tr = \tilde{\Theta}(nk)$. Additionally, we show the same efficiency may be achieved with a single-round query algorithm using batch codes. We leave the following high-level open question:

What techniques may be combined with private preprocessing to further improve the efficiency of PIR?

In this work, we ruled out using batch PIR techniques to further speed up PIR with private preprocessing. One way to improve PIR efficiency may be use more complex encodings of databases beyond replication.

Acknowledgements. The author would like to thank Chengyu Lin, Zeyu Liu and Tal Malkin for initial discussions and feedback on earlier versions of this paper, Giuseppe Persiano for reading an early draft and helpful suggestions for improving the paper, Henry Corrigan-Gibbs, Alexandra Henzinger and Dmitry Kogan for discussion about the pipelined queries approach and Josh Alman for discussion on the online matrix-vector conjecture. This research was supported

in part by the Algorand Centres of Excellence programme managed by Algorand Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are solely those of the authors.

References

1. H. Abusalah, J. Alwen, B. Cohen, D. Khilko, K. Pietrzak, and L. Reyzin. Beyond hellman's time-memory trade-offs with applications to proofs of space. In *ASIACRYPT*, 2017.
2. C. Aguilar-Melchor, J. Barrier, L. Fousse, and M.-O. Killijian. XPIR: Private information retrieval for everyone. *PoPETS*, 2016.
3. A. Ali, T. Lepoint, S. Patel, M. Raykova, P. Schoppmann, K. Seth, and K. Yeo. Communication–computation trade-offs in PIR. In *USENIX Security*, 2021.
4. A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *ICALP*, 1997.
5. S. Angel, H. Chen, K. Laine, and S. Setty. PIR with compressed queries and amortized query processing. In *IEEE symposium on security and privacy*, 2018.
6. S. Angel and S. Setty. Unobservable communication over fully untrusted infrastructure. In *USENIX OSDI*, 2016.
7. A. Beimel and Y. Ishai. Information-theoretic private information retrieval: A unified construction. In *ICALP*, 2001.
8. A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *ACM STOC*, 1999.
9. A. Beimel, Y. Ishai, E. Kushilevitz, and J.-F. Raymond. Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. *FOCS*, 2002.
10. A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *Journal of cryptology*, 2004.
11. N. Borisov, G. Danezis, and I. Goldberg. DP5: A private presence service. *Proc. Priv. Enhancing Technol.*, 2015.
12. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. In *EUROCRYPT*, 2015.
13. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing: Improvements and extensions. In *ACM SIGSAC CCS*, 2016.
14. E. Boyle, J. Holmgren, and M. Weiss. Permuted puzzles and cryptographic hardness. In *Theory of Cryptography Conference*, 2019.
15. E. Boyle, Y. Ishai, R. Pass, and M. Wootters. Can we access a database both locally and privately? In *Theory of Cryptography Conference*, 2017.
16. C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, 1999.
17. R. Canetti, J. Holmgren, and S. Richelson. Towards doubly efficient private information retrieval. In *Theory of Cryptography Conference*, 2017.
18. R. Cheng, W. Scott, B. Parno, I. Zhang, A. Krishnamurthy, and T. Anderson. Talek: a private publish-subscribe protocol. Technical report, 2016.
19. B. Chor and N. Gilboa. Computationally private information retrieval. In *STOC*, 1997.
20. B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM*, 1998.
21. S. Coretti, Y. Dodis, S. Guo, and J. Steinberger. Random oracles and non-uniformity. In *EUROCRYPT*, 2018.

22. H. Corrigan-Gibbs, A. Henzinger, and D. Kogan. Single-server private information retrieval with sublinear amortized time. *EUROCRYPT*, 2022.
23. H. Corrigan-Gibbs and D. Kogan. The discrete-logarithm problem with preprocessing. In *EUROCRYPT*, 2018.
24. H. Corrigan-Gibbs and D. Kogan. Private information retrieval with sublinear online time. In *EUROCRYPT*, 2020.
25. A. Davidson, G. Pestana, and S. Celi. Frodopir: Simple, scalable, single-server private information retrieval. In *PETS*, 2023.
26. A. De, L. Trevisan, and M. Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In *CRYPTO*, 2010.
27. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for private information retrieval. *Journal of Cryptology*, 2001.
28. G. Di Crescenzo, T. Malkin, and R. Ostrovsky. Single database private information retrieval implies oblivious transfer. In *EUROCRYPT*, 2000.
29. Y. Dodis, S. Guo, and J. Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In *EUROCRYPT*, 2017.
30. Z. Dvir, P. Gopalan, and S. Yekhanin. Matching vector codes. In *FOCS*, 2010.
31. Z. Dvir and S. Gopi. 2-server PIR with subpolynomial communication. *JACM*, 2016.
32. K. Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 2012.
33. R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM Journal on Computing*, 2005.
34. C. Gentry and S. Halevi. Compressible FHE with applications to PIR. In *TCC*, 2019.
35. C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *ICALP*, 2005.
36. N. Gilboa and Y. Ishai. Distributed point functions and their applications. In *EUROCRYPT*, 2014.
37. O. Goldreich, H. Karloff, L. J. Schulman, and L. Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. In *CCC*, 2002.
38. M. Green, W. Ladd, and I. Miers. A protocol for privately reporting ad impressions at scale. In *ACM SIGSAC CCS*, 2016.
39. J. Groth, A. Kiayias, and H. Lipmaa. Multi-query computationally-private information retrieval with constant communication rate. In *PKC*, 2010.
40. T. Gupta, N. Crooks, W. Mulhern, S. Setty, L. Alvisi, and M. Walfish. Scalable and private media consumption with Popcorn. In *USENIX NSDI*, 2016.
41. A. Hamlin, R. Ostrovsky, M. Weiss, and D. Wichs. Private anonymous data access. In *EUROCRYPT*, 2019.
42. R. Henry. Polynomial batch codes for efficient IT-PIR. *PoPETS*, 2016.
43. R. Henry, Y. Huang, and I. Goldberg. One (block) size fits all: PIR and SPIR with variable-length records via multi-block queries. In *NDSS*, 2013.
44. A. Henzinger, M. M. Hong, H. Corrigan-Gibbs, S. Meiklejohn, and V. Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In *USENIX Security*, 2023.
45. M. Henzinger, S. Krinninger, D. Nanongkai, and T. Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *STOC*, 2015.
46. P. Hubáček, M. Koucký, K. Král, and V. Slívová. Stronger lower bounds for online ORAM. In *TCC*, 2019.

47. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. In *ACM symposium on Theory of computing*, 2004.
48. R. Jacob, K. G. Larsen, and J. B. Nielsen. Lower bounds for oblivious data structures. In *SODA*, 2019.
49. D. Kogan and H. Corrigan-Gibbs. Private blocklist lookups with checklist. In *USENIX Security*, 2021.
50. I. Komargodski and W.-K. Lin. A logarithmic lower bound for oblivious ram (for all parameters). In *CRYPTO*, 2021.
51. E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, 1997.
52. E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In *EUROCRYPT*, 2000.
53. K. G. Larsen. The cell probe complexity of dynamic range counting. In *STOC*, 2012.
54. K. G. Larsen and J. B. Nielsen. Yes, there is an oblivious RAM lower bound! In *CRYPTO*, 2018.
55. K. G. Larsen and R. Williams. Faster online matrix-vector multiplication. In *SODA*, 2017.
56. W. Lueks and I. Goldberg. Sublinear scaling for multi-client private information retrieval. In *FC*, 2015.
57. P. Mittal, F. Olumofin, C. Troncoso, N. Borisov, and I. Goldberg. PIR-Tor: Scalable anonymous communication using private information retrieval. In *USENIX Security Symposium*, 2011.
58. M. H. Mughees, H. Chen, and L. Ren. OnionPIR: Response efficient single-server PIR. In *ACM SIGSAC CCS*, 2021.
59. R. Ostrovsky and W. E. Skeith. A survey of single-database private information retrieval: Techniques and applications. In *PKC*, 2007.
60. S. Patel, G. Persiano, and K. Yeo. Private stateful information retrieval. In *ACM SIGSAC CCS*, 2018.
61. S. Patel, G. Persiano, and K. Yeo. Lower bounds for encrypted multi-maps and searchable encryption in the leakage cell probe model. In *CRYPTO*, 2020.
62. M. B. Paterson, D. R. Stinson, and R. Wei. Combinatorial batch codes. *Advances in Mathematics of Communications*, 2009.
63. G. Persiano and K. Yeo. Lower bounds for differentially private RAMs. In *Eurocrypt*, pages 404–434. Springer, 2019.
64. G. Persiano and K. Yeo. Limits of preprocessing for single-server PIR. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, 2022.
65. G. Persiano and K. Yeo. Lower bound framework for differentially private and oblivious data structures. In *EUROCRYPT*, 2023.
66. S. Servan-Schreiber, K. Hogan, and S. Devadas. Adveil: A private targeted-advertising ecosystem. *Cryptology ePrint Archive*, 2021.
67. E. Shi, W. Aqeel, B. Chandrasekaran, and B. Maggs. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In *CRYPTO*, 2021.
68. D. Unruh. Random oracles and auxiliary input. In *CRYPTO*, 2007.
69. S. Wehner and R. d. Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In *ICALP*, 2005.
70. S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM*, 2008.
71. K. Yeo. Lower bounds for (batch) pir with private preprocessing. Cryptology ePrint Archive, Paper 2022/828, 2022. `https://eprint.iacr.org/2022/828`.