Batch Private Information Retrieval 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 in t online time. For privacy, the client wishes to hide index i from an adversary that has compromised some of the servers. We will focus on the *batch PIR* setting where the client performs queries to retrieve the contents of multiple entries simultaneously.

We present a tight characterization for the trade-offs between hint size and online query time. For any $\ell = O(1)$ and ℓ-server 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 server. Our results also improve the best known lower bounds for the single query setting by a logarithmic factor. On the positive side, we show there exists a construction with a single-round query algorithm such that $tr = \tilde{O}(nk)$ that matches our lower bound up to logarithmic factors.

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 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 [4,19,7,9,66,31,30] and references therein). Kushilevitz and Ostrovsky [47] first studied PIR in the single-server setting against computationally bounded adversaries. Many further works have also studied single-server PIR including [16,8,27,48,34,54,2,5,33,55,3,53] to list some examples.

PIR is an important cryptographic tool due to its endless implications to real-world settings. As some examples, PIR could enable users to access information about medical or financial data without revealing their interests to the data provider that could include the queried health condition or investment instrument. PIR has been used as a critical component in the design of many practical privacy-preserving applications such as advertising [37,61], communication [52,6], friend discovery [11], media consumption [39] 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 prohibits 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. This phenomenon is easy to see in the single-server setting where, if a single database entry is not processed during query time, the adversarial server can determine the user's lack of interest in the untouched entry.

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 $O(n^{1/2+\epsilon})$ query time 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 [60]. 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 in the public preprocessing setting.

As an analog, one can also consider the *private preprocessing* setting (also known as offline-online PIR) where the r-bit hint H is computed and stored by the client hidden from the adversary's view. This model has been studied in many works including [26,15,17,56,14,40,62,22]. Corrigan-Gibbs and Kogan [24] presented a tight trade-off between the hint size r and online server time t of $tr = \tilde{\Theta}(n)$. Note, 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. In the rest of the paper, we will focus on the private preprocessing model as it enables efficient constructions with more immediate implications towards practical applications.

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 [10,44,58,38,42,51,6,5]. 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 [44] showed the existence of a batch PIR that uses total server time $t = \tilde{O}(n)$ or amortized query time of $\tilde{O}(n/k)$ when retrieving k entries simultaneously. Furthermore, prior works [6,5] have also presented probabilistic batch PIR constructions aiming for concrete efficiency with similar asymptotic overhead, but only have experimental analysis to bound error probabilities.

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 [24,62] or the single-server scheme in [22]). 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 online query time for PIR schemes that aim to combine batching and offline private preprocessing techniques. To do this, we will present a lower bound that encompasses a wide range of constructions and matches the overhead of prior works. Additionally, we show it is possible to build an optimal single-round query construction matching our lower bound.

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 server time during online query execution, t. 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. We refer readers to Section 2.2 for more details on the lower bound model. We present our lower bound below:

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. In general, a construction may ignore the hint and execute a k-query batch PIR using $\tilde{O}(n)$ time. In the case that $r < k$, this is the optimal construction matching our lower bound. There is another 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 $c \leq 400$.

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.

Discussion about Single Query Lower Bounds. We note that one can obtain a single query lower bound from our results by setting $k = 1$ to obtain that $tr = \Omega(n)$. To our knowledge, this is logarithmically higher than the previous known single query lower bounds of $tr = \tilde{\Omega}(n)$ [24,22]. Prior works proved lower bounds through a reduction to Yao's Box problem [65] that seems to result in a slightly lower bound when generalizing to non-zero error probabilities using presampling [63,21]. By proving our lower bound directly through an incompressibility argument, we are able to avoid the loss of a multiplicative logarithmic factor.

Discussion about Independent Work. In an concurrent work, Corrigan-Gibbs, Henzinger and Kogan [22] proved a similar lower bound of $tr = \tilde{\Omega}(nk)$ through a reduction to a multi-box generalization of Yao's Box problem [65]. We note the lower bound proved in our work is also higher by a multiplicative logarithmic factor compared to the lower bound in [22].

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. We show that there exists constructions that also match our lower bound where the query algorithm uses only a single-round of interaction.

Theorem 2 (Informal). *Assuming the existence of a single-query PIR with private preprocessing with $tr = f(n)$, there exists a k-query batch PIR with private preprocessing scheme with a single-round query such that $tr = \tilde{O}(k \cdot f(n))$.*

Discussion about Pipelined Queries. Prior works presented single query schemes that enjoy a property referred to as malicious security [24,62,46,22]. This property ensures that each client query is independent of server responses. For any construction that supports multiple queries with a single preprocessing stage, the client may issue multiple queries simultaneously without waiting for responses to prior queries. This is referred to as *pipelined queries* in [46]. At first, this seems to enable the client to issue k queries in parallel to construct a single round optimal batch construction. Unfortunately, this approach does not work for arbitrary values of r and k. We refer readers to Appendix A for more details.

1.2 Technical Overview

We present an overview of the techniques that are used for our lower bound proof in Section 3 and our single-round query construction in Section 4.

Lower Bound. The core idea of our lower bound 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 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., $|P \cap q| \leq k/2$). In our main proof, 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 t 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.

We note that our proof avoids a reduction to Yao's Box problem [65]. Instead, we go through an incompressibiliy argument of a random database directly. As a result, we prove logarithmically higher lower bounds than prior works [24,22].

Upper Bound. For our single-round construction, we will leverage batch codes (introduced by Ishai, Kushilevitz, Ostrovsky and Sahai [44]) 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 [44,6,5] 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 $tr = \widetilde{O}(n)$ matching our lower bound.

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 [4,7,9,66,31]. The most communication-efficient scheme is by Dvir and Gopi [30] using matching vector codes [29]. Similar work has been done for computationally-secure multi-server PIR [19,35,12] where the most efficient constructions utilize function secret sharing techniques [13]. Single-server PIR was introduced by Kushilevitz and Ostrovsky [47] with many follow-up works including [16,8,27,48,34,54] 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 [2,5,33,55,3,53].

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 [15,60]. 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 [56], private anonymous data access [40] and offline-online PIR [24,62]. For private preprocessing, Corrigan-Gibbs and Kogan [24] presented a construction with optimal trade-offs $tr = \widetilde{O}(n)$ that was later extended to handle blocklist lookups efficiently [46]. Shi, Aqeel, Chandrasekaran and Maggs [62] presented logarithmic communication two-server schemes with optimal trade-offs. Finally, Corrigan-Gibbs, Henzinger and Kogan [22] presented single-server constructions with optimal trade-offs where a single preprocessing stage may be used for multiple queries.

Batch PIR. Several prior works have studied batch PIR to obtain efficient constructions using matrix multiplication [10,51], batch codes [44,58,41], the ϕ-hiding assumption [38] and list-decoding algorithms [42]. 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 [36,64].

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 [60]. 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] showed a tight lower bound of $tr = \widetilde{\Omega}(n)$ for a single query. The follow-up work of Corrigan-Gibbs, Henzinger and Kogan [22] showed the same lower bound holds even when considering schemes that enable multiple single-entry PIR retrievals in sequence.

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 [24,60,22]. To list some examples outside of PIR, incompressibility has been used in the studies of generic cryptographic constructions [32], one-way functions and PRGs [25], proofs of space [1], random oracles [63,28,21], the discrete logarithm problem [23] and oblivious data structures [50,45,59,43,57].

2 Definitions

2.1 Batch PIR with Private Preprocessing

We start by defining batch PIR with private preprocessing (also known 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). *A k-query batch PIR with private preprocessing scheme $\mathit{\Pi} = (\mathit{\Pi}.\mathsf{Preprocess}, \mathit{\Pi}.\mathsf{Encode}, \mathit{\Pi}.\mathsf{Query})$ is in the standard PIR model if it satisfies Definition 1 and $\mathit{\Pi}.\mathsf{Encode}$ is the identity function.*

Constructions in the standard PIR model ensure that the server stores the database without encoding. We will use the standard PIR model for our lower bound and point readers to Section 2.2 for further discussion. We will use the notion of encoding algorithms to present our constructions in a natural manner.

Next, we will define the correctness of constructions. We define a query as correct if all 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 $\mathit{\Pi}$ errs with probability at most ϵ if, for every database $D \in \{0,1\}^n$ and query $q \subseteq [n]$, it holds that*

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

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

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 following game:

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

1. The challenger $\mathcal{C}$ runs $E \leftarrow \mathit{\Pi}.\mathsf{Encode}(D)$.
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 $H \leftarrow \mathit{\Pi}.\mathsf{Preprocess}(R_H; D)$ to obtain hint H using random coin tosses R_H and records transcripts $\mathcal{T}^p_1, \ldots, \mathcal{T}^p_\ell$.
4. The challenger $\mathcal{C}$ executes $\mathit{\Pi}.\mathsf{Query}(q^\eta, H, R; E)$ 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, $\{\mathcal{T}^p_i, \mathcal{T}_i\}_{i \in S}$, to the adversary $\mathcal{A}$.
6. The adversary $\mathcal{A}(\{\mathcal{T}^p_i, \mathcal{T}_i\}_{i \in S})$ outputs a bit b.

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:*

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

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 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 [10,60]).

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

We will use the standard PIR model that has been used to prove PIR lower bounds in prior works [10,24,60,22]. In the standard PIR model, the database is stored by the server without encoding. Additionally, the client is able to store an 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. Note we only measure costs using server operations. In other words, 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.

In our model, we assume constructions use finite, but arbitrarily long, randomness for the constructions. Note, this does not rule out algorithms that may use infinite randomness that decrease the error probability with longer running time and/or randomness usage. Instead, we can convert such an algorithm into another one that terminates once the error probability is small enough after using arbitrary long, but finite, randomness and provides an answer.

Finally, we comment on the encoding restriction of the lower bound model. In practice, the lack of database encoding ends up making constructions easier to implement and use. Additionally, 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 $\tilde{\Omega}(\log^2 n)$ [49] that are too low to prove meaningful lower bounds for PIR currently. We also note that proving similar lower bounds as ours with arbitrary server encoding 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.

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 3. *Let Π be a k-batch, ℓ-server PIR with private preprocessing scheme in the standard PIR model 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 4. *Let Π be a k-batch, ℓ-server PIR with private preprocessing scheme in the standard PIR model 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 4 immediately implies Theorem 3.

Proof of Theorem 3. 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 3. 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 4.

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 4. $\square$

Proof Overview. The rest of this section will be devoted to prove Theorem 4. Our proof of Theorem 4 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 an assumption towards a contradiction. Suppose there exists a too-good-to-be-true Π such that $tr = o(nk)$. We will assume that $t \leq n/(100\ell)$. 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 we assumed that $r \geq k$ 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 $i \in S$, it must be that*

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

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 $t \leq n/(100\ell)$. □

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

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

– Challenge Phase $\mathcal{A}_{i,q}(D)$:
    1. Return $(\{1, \dots, k\}, q, \{x\})$ 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 $\{1, \dots, k\}$ 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 $\{1, \dots, k\}$, 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. □

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 $q = \{i_1, \dots, i_k\}$ be the query to k indices $i_1, \dots, i_k$. 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 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

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

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.

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 the correct database contents of each queried entry, 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 condition 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 (1/5)-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 satify 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

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

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:

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

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 S. 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 □

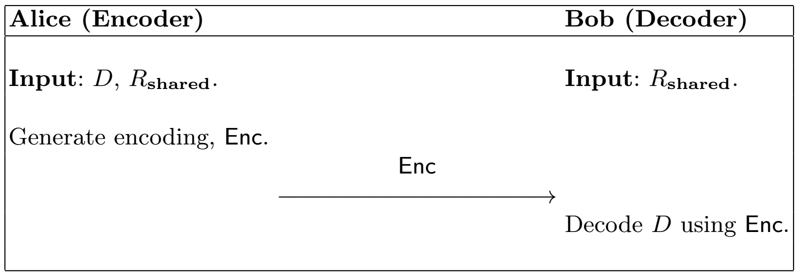
3.3 An Impossible Encoding

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 the 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 on the input database D. We present a diagram of the one-way communication protocol setting below.


Alice (Encoder)
Bob (Decoder)
Input: D, $R_{\mathbf{shared}}$.
Input: $R_{\mathbf{shared}}$.
Generate encoding, Enc.
$\xrightarrow{\mathsf{Enc}}$
Decode D using Enc.

Next, we prove a lemma that shows that the length of the Alice's encoding cannot be much smaller than the minimum number of random bits stored in a database drawn from the hard distribution.

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 $\mathbf{R}_{\mathbf{shared}}$ and any other 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,

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

The first equality is from the fact $\mathbf{R}_{\mathbf{shared}}$ 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 described the shared randomness $\mathbf{R}_{\mathbf{shared}}$. In particular, $\mathbf{R}_{\mathbf{shared}}$ 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 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

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

```
1. Set $H \leftarrow \mathit{\Pi}.\mathsf{Preprocess}(R_H; D)$ 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**.) $\mathit{\Pi}.\mathsf{Query}(q_i, H, R; D) = (D_x)_{x \in q_i}$.
       ii. (**Overlap with Known Entries.**) $|q_i \cap A| \leq k/10$.
       iii. (**Overlap with Probes.**) $|q_i \cap \mathit{\Pi}.\mathsf{Probes}(q_i, D, H, R)| \leq 4k/5$.
   (c) If $q_i$ does satisfies the above:
       i. Set $U \leftarrow \mathit{\Pi}.\mathsf{Probes}(q_i, D, H, R) \setminus A$ 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 $\mathsf{Enc}$.
```

Fig. 1. 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. 2. Description of Bob's decoding algorithm.

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. A formal description of Alice's encoding algorithm is provided in Figure 1.

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. A formal description of Bob's decoding algorithm is provided in Figure 2.

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 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

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

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

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

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

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

where we use Lemma 4 to bound $\Pr_{\mathbf{H},\mathbf{R}}[E^R(\mathbf{H},\mathbf{R}) = 0] \leq 49/50$ and the fact that $f < 1$.

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, we know that Alice's encoding length is at most

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

using the fact that $\binom{a}{b} \leq (ea/b)^b$.

Let p be the probability that Alice's encoding starts with a 0-bit where we know that p is a constant and $p < 1$. Then, we get that Alice's expected encoding length is at most

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

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

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

where we will assume that $k \geq 20 \log(2000e)$ and use the fact that $(1-p) > 0$ as $p < 1$. Finally, we use the facts that p is constant and $r \leq n/400$ to see that $rk/10 \leq nk/4000$ and $1/(1-p) = \Theta(1)$ and get that

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

where we also use that $r \geq k$. We can assume that n is sufficiently large to get that $tr = \Omega(nk)$ to complete the proof. □

4 Upper Bound

In this section, we present upper bounds for batch PIR with private preprocessing. First, we show that there already exists a naive way to modify prior PIR with private preprocessing constructions to obtain constructions matching our lower bound up to logarithmic factors. However, these constructions require $O(k)$-round query algorithms. Afterwards, we present constructions with single-round queries that obtain similar efficiency using batch codes.

4.1 Multiple Round Constructions

We can adapt any constructions with the feature that a single hint may be used for multiple queries to obtain tight constructions. That is, a single preprocessing stage may be followed by multiple queries. In particular, we can leverage the two-server schemes in [24,62] or the single-server scheme in [22] that have the necessary feature. These constructions use $t = \tilde{O}(n/r)$ time when using r-bit hints. However, the online query time includes any necessary refreshing of the hint for future queries.

By performing k sequential queries, we get a construction with $tr = \tilde{O}(nk)$ that requires k rounds of client-server interaction. This is exactly one of the straightforward approaches outlined in Section 1. As a result, we can obtain the following construction:

Theorem 5. *Assuming the existence of a single-query PIR with private preprocessing scheme that is $(\delta, 1, \ell)$-secure such that $tr = \tilde{O}(n)$, can handle multiple queries with a single preprocessing stage and is correct except with negligible probability, then there exists a single-query PIR with private preprocessing scheme that is $(k\delta, 1, \ell)$-secure, uses r-bit hints and online query time t such that $tr = \tilde{O}(nk)$ and returns the correct answer except with negligible probability. The online query algorithm requires k rounds of interaction.*

As the constructions in [24,62,22] satisfy the requirement, we obtain a construction that matches our lower bound (up to logarithmic factors) as our lower bound does not assume anything about the interactivity of the query algorithm.

4.2 Single-Round Constructions

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 [44,6,5] 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 [44] 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

```
Preprocess($D$):
  1. Compute $C(D) = (B_1, \ldots, B_m)$.
  2. For all $i = 1, \ldots, m$:
     (a) Execute $H_i \leftarrow \Pi.\mathsf{Preprocess}(B_i)$.
  3. Return $(H_1, \ldots, H_m)$.

Encode($D$):
  1. Compute $C(D) = (B_1, \ldots, B_m)$.
  2. Return $(B_1, \ldots, B_m)$.

Query($\{i_1, \ldots, i_k\}, (H_1, \ldots, H_m); (B_1, \ldots, B_m)$):
  1. Compute $C_A(\{i_1, \ldots, i_k\}) = ((a_1, b_1), \ldots, (a_k, b_k))$.
  2. For $x = 1, \ldots, m$:
     (a) If $x = a_j$ for some $j \in [k]$:
         i. Execute $D_{i_j} \leftarrow \Pi.\mathsf{Query}(H_x, b_j; B_x)$.
     (b) Otherwise:
         i. Execute $\Pi.\mathsf{Query}(H_x, 1; B_x)$ and ignore result.
  3. Return $(D_{i_1}, \ldots, D_{i_k})$.
```

Fig. 3. Description of our batch PIR with private preprocessing construction where C is a $(n, N, k, m, 1)$ systematic batch code and Π is a single-query PIR with private preprocessing scheme.

from each of the m buckets. These codes have been referred to as replication-based batch codes [44] or combinatorial batch codes [58] 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 (see [44,6,5] as an example).

Definition 6 ([58]). *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 $(j_1, \ldots, j_k)$ 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. We describe our construction formally in Figure 3.

Theorem 6. *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.

Proof. By properties of the batch code, we know that our construction will always be able to identify k buckets that contain the k queried entries. By the properties of Π, we know that that the probability that any of these k queries result in the wrong answer is at most k times the probability that Π returns the wrong answer resulting in error probability $k\epsilon$. Finally, as the transcript seen by the adversarial server is identical to the transcript seen by executing Π m parallel times, we know that our scheme is $(m\delta, \ell_{\mathcal{A}}, \ell)$-secure when Π is $(\delta, \ell_{\mathcal{A}}, \ell)$-secure.

For efficiency, note that each of the m instances is allocated r'/m bits for its own hint. As the database for the i-th bucket contains N_i entries, we know that the i-th instance of Π requires $t(N_i, r'/m)$ online query time. So, we see that the online query time for our scheme is $t'(n, r') = t(N_1, r'/m) + \ldots + t(N_m, r'/m)$. $\square$

Relation to Lower Bound Model. Note that our lower bound model assumes that there is no encoding of the database while our construction encodes the database using a systematic batch code. We chose to present our construction in Figure 3 as the most efficient and natural way to use batch codes as opposed to one that fit into our lower bound model. However, it is straightforward to modify the construction such that the server does not encode the database and, thus, is compatible with our lower bound model. Each time the query algorithm needs to probe one of the entries in the m buckets, the client will also encode the original database entry that should appear in that bucket entry. This increases the communication cost of our schemes. However, both the hint size r and online query server time t remain identical. As a result, our construction may be trivially modified to be compatible with our lower bound model without affecting the hint size to server query time trade-off.

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 [24,62,22] to get the following constructions from batch codes.

Theorem 7. *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.*

Proof. We can apply Theorem 6 to get the security and correctness properties. For efficiency, let $N_1, \ldots, N_m$ be the number of entries encoded in each of the m buckets. Then, we know that

$$t = \tilde{O}(N_1 \cdot m/r) + \tilde{O}(N_m \cdot m/r) = \tilde{O}(N \cdot m/r)$$

to complete the proof. □

For instantiating batch codes, many prior works (see [44,58,41,6,5] 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 [44] 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 [44] batch code to obtain an explicit batch PIR with private preprocessing. However, these explicit constructions will result in worse parameters.

Practical Batch Codes with Experimental Analysis. In terms of practical constructions for batch codes, several prior works [6,5] have presented schemes that aim for concrete efficiency with only experimental analysis. In particular, both papers present *probabilisitic batch codes* from hashing schemes that result in $O(n)$ total codeword size and $O(k)$ buckets. In other words, these are the batch codes with the best possible asymptotic parameters of $(n, O(n), k, O(k), 1)$. Plugging such a batch code into our scheme would result in a construction matching our lower bound of $tr = \tilde{O}(nk)$. However, these constructions have the property that subset of queries will not be able to decode and always err. Note, this is different from the natural notion of error probabilities considered in our work where the probability is over the internal randomness of the algorithm. Additionally, to our knowledge, the only error analysis are done through experimental evaluation. We are unaware of a theoretical analysis bounding the error of these constructions necessary for our needs. Regardless, these constructions probably remain the best approach for constructions that aim for concrete efficiency and practical application.

5 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. In terms of open problems, we leave the following high-level 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.

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. The author would also like to thank Giuseppe Persiano for reading an early draft and helpful suggestions for improving the paper.

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 *International Conference on the Theory and Application of Cryptology and Information Security*, pages 357–379. Springer, 2017.
2. C. Aguilar-Melchor, J. Barrier, L. Fousse, and M.-O. Killijian. XPIR: Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies*, 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 Symposium*, 2021.

4. A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *International Colloquium on Automata, Languages, and Programming*, 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 Symposium on Operating Systems Design and Implementation*, 2016.
7. A. Beimel and Y. Ishai. Information-theoretic private information retrieval: A unified construction. In *International Colloquium on Automata, Languages, and Programming*, 2001.
8. A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *ACM symposium on Theory of computing*, 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 *Annual international conference on the theory and applications of cryptographic techniques*, 2015.
13. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing: Improvements and extensions. In *ACM SIGSAC Conference on Computer and Communications Security*, 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 *International Conference on the Theory and Applications of Cryptographic Techniques*, 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, Technical Report UW-CSE-16-11-01, University of Washington Computer Science, 2016.
19. B. Chor and N. Gilboa. Computationally private information retrieval. In *ACM symposium on Theory of computing*, 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 *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 227–258. Springer, 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 *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–447. Springer, 2018.
24. H. Corrigan-Gibbs and D. Kogan. Private information retrieval with sublinear online time. In *Eurocrypt*, 2020.

25. A. De, L. Trevisan, and M. Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In *Annual Cryptology Conference*, pages 649–665. Springer, 2010.
26. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for private information retrieval. *Journal of Cryptology*, 2001.
27. G. Di Crescenzo, T. Malkin, and R. Ostrovsky. Single database private information retrieval implies oblivious transfer. In *International Conference on the Theory and Applications of Cryptographic Techniques*, 2000.
28. Y. Dodis, S. Guo, and J. Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 473–495. Springer, 2017.
29. Z. Dvir, P. Gopalan, and S. Yekhanin. Matching vector codes. In *Annual Symposium on Foundations of Computer Science*, 2010.
30. Z. Dvir and S. Gopi. 2-server PIR with subpolynomial communication. *Journal of the ACM*, 2016.
31. K. Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 2012.
32. R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM journal on Computing*, 35(1):217–246, 2005.
33. C. Gentry and S. Halevi. Compressible FHE with applications to PIR. In *Theory of Cryptography Conference*, 2019.
34. C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *International Colloquium on Automata, Languages, and Programming*, 2005.
35. N. Gilboa and Y. Ishai. Distributed point functions and their applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2014.
36. O. Goldreich, H. Karloff, L. J. Schulman, and L. Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. In *IEEE Annual Conference on Computational Complexity*, 2002.
37. M. Green, W. Ladd, and I. Miers. A protocol for privately reporting ad impressions at scale. In *ACM SIGSAC Conference on Computer and Communications Security*, 2016.
38. J. Groth, A. Kiayias, and H. Lipmaa. Multi-query computationally-private information retrieval with constant communication rate. In *International Workshop on Public Key Cryptography*, 2010.
39. T. Gupta, N. Crooks, W. Mulhern, S. Setty, L. Alvisi, and M. Walfish. Scalable and private media consumption with Popcorn. In *USENIX Symposium on Networked Systems Design and Implementation*, 2016.
40. A. Hamlin, R. Ostrovsky, M. Weiss, and D. Wichs. Private anonymous data access. In *Eurocrypt*, 2019.
41. R. Henry. Polynomial batch codes for efficient IT-PIR. *Proceedings on Privacy Enhancing Technologies*, 2016.
42. 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.
43. P. Hubáček, M. Kouckỳ, K. Král, and V. Slívová. Stronger lower bounds for online ORAM. In *Theory of Cryptography Conference*, pages 264–284. Springer, 2019.
44. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. In *ACM symposium on Theory of computing*, 2004.

45. R. Jacob, K. G. Larsen, and J. B. Nielsen. Lower bounds for oblivious data structures. In *SODA*, pages 2439–2447. SIAM, 2019.
46. D. Kogan and H. Corrigan-Gibbs. Private blocklist lookups with checklist. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 875–892, 2021.
47. E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, 1997.
48. E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In *International Conference on the Theory and Applications of Cryptographic Techniques*, 2000.
49. K. G. Larsen. The cell probe complexity of dynamic range counting. In *ACM symposium on Theory of computing*, 2012.
50. K. G. Larsen and J. B. Nielsen. Yes, there is an oblivious RAM lower bound! In *Annual International Cryptology Conference*, pages 523–542. Springer, 2018.
51. W. Lueks and I. Goldberg. Sublinear scaling for multi-client private information retrieval. In *International Conference on Financial Cryptography and Data Security*, 2015.
52. 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.
53. M. H. Mughees, H. Chen, and L. Ren. OnionPIR: Response efficient single-server PIR. In *ACM SIGSAC Conference on Computer and Communications Security*, 2021.
54. R. Ostrovsky and W. E. Skeith. A survey of single-database private information retrieval: Techniques and applications. In *International Workshop on Public Key Cryptography*, 2007.
55. J. Park and M. Tibouchi. SHECS-PIR: somewhat homomorphic encryption-based compact and scalable private information retrieval. In *European Symposium on Research in Computer Security*, 2020.
56. S. Patel, G. Persiano, and K. Yeo. Private stateful information retrieval. In *ACM SIGSAC Conference on Computer and Communications Security*, 2018.
57. S. Patel, G. Persiano, and K. Yeo. Lower bounds for encrypted multi-maps and searchable encryption in the leakage cell probe model. In *Advances in Cryptology – CRYPTO 2020*, 2020.
58. M. B. Paterson, D. R. Stinson, and R. Wei. Combinatorial batch codes. *Advances in Mathematics of Communications*, 2009.
59. G. Persiano and K. Yeo. Lower bounds for differentially private RAMs. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 404–434. Springer, 2019.
60. G. Persiano and K. Yeo. Limits of preprocessing for single-server PIR. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, 2022.
61. S. Servan-Schreiber, K. Hogan, and S. Devadas. Adveil: A private targeted-advertising ecosystem. *Cryptology ePrint Archive*, 2021.
62. E. Shi, W. Aqeel, B. Chandrasekaran, and B. Maggs. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In *Annual International Cryptology Conference*, 2021.
63. D. Unruh. Random oracles and auxiliary input. In *Annual International Cryptology Conference*, pages 205–223. Springer, 2007.
64. S. Wehner and R. d. Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In *International Colloquium on Automata, Languages, and Programming*, 2005.

65. A.-C. Yao. Coherent functions and program checkers. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 84–94, 1990.
66. S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM*, 2008.

A Discussion about Pipelined Queries Approach

We start by summarizing the constructions in [24,46,62] that enable multiple queries for a single preprocessing stage in the two-server setting. In the offline phase, the client will generate r pseudorandom subsets $S_1, \ldots, S_r \subset [n]$ each of size t. These will be sent to the first (offline) server that will return the r parties of the form $b_i = \oplus_{s \in S_i} D_s$ for database D. The hint becomes $b_1, \ldots, b_r$. To query for any index i, the client first finds a set S_j such that $i \in S_j$. With probability approximately $1 - 1/\sqrt{n}$, the client uploads $S_j \setminus \{i\}$ to the second (online) server that computes and returns $b = \oplus_{s \in S_j \setminus \{i\}} D_s$. Otherwise, the client removes a random index from S_j and will fail to retrieve the desired entry D_i correctly. Finally, the client retrieves $D_i = b \oplus b_j$. With parallel repetition, the error probability can be driven down to negligible. Prior works [24,46,62] have shown that this is secure and can be extended to refresh the bit b_j with $O(t)$ time such that more queries can be handled without re-running the preprocessing stage. Here, the pipelined query property refers to the fact that the client can issue multiple queries simultaneously without waiting for server responses.

Initially, the pipelined queries approach seems to enable building a single round batch PIR with private preprocessing scheme from a single PIR with private preprocessing scheme. The client may issue all k queries in parallel using a single round. However, this approach requires that for each query i in the batch, the client must find a unique subset S_j such that $i \in S_j$. While this is highly likely for a single query, it is not always possible when considering batches of k queries for certain values of r.

Consider a concrete setting where $k = n^{1/2}$ and $r = \tilde{O}(n^{1/2})$ and the goal is to construct a scheme with $tr = \tilde{O}(nk)$. As we need to issue $k = n^{1/2}$ queries, we require that the single query scheme must have online time $\tilde{O}(n^{1/2})$ to guarantee that $tr = \tilde{O}(nk$. In the offline phase, the client will retrieve r parities of pseudorandom subsets $S_1, \ldots, S_r$ of size $\tilde{O}(n^{1/2})$. For the single query setting, the client may perform a query as long as the query index i appears in $S_1 \cup \ldots \cup S_r$. For a batch of k queries $\{i_1, \ldots, i_k\}$, the client must find k different subsets $S_{j_1}, \ldots, S_{j_k}$ for each of the k queries such that $i_x \in S_{i_x}$ to be able to send k batch queries in parallel. We sketch that this occurs only with very low probability for our chosen values of r and k resulting in very high error probability.

Pick a random batch query set of size $k = n^{1/2}$. Suppose we are lucky enough that the client successfully found $n^{1/2} - n^{1/8}$ different subsets for the first $n^{1/2} - n^{1/8}$ indices in the batch query. We will consider the remaining $n^{1/8}$ queries. Note, there remain only $r - (n^{1/2} - n^{1/8}) = \tilde{O}(n^{1/8})$ subsets that may be used covering a total of $\tilde{O}(n^{5/8})$ elements from $[n]$. As a result, the probability that any random index appears in the remaining valid subsets is approximately

$\tilde{O}(n^{5/8}/n) = \tilde{O}(n^{-3/8})$. The probability that all remaining $n^{1/8}$ random elements are covered is at most $n^{-\Theta(n^{1/8})}$. In other words, the probability that the pipelined queries approach would result in correctly retrieving all k queries in a single round would be at most $n^{-\Theta(n^{1/8})}$, which is exponentially small.

In the above example, we have shown that the pipelined approach cannot construct batch schemes with reasonable error probabilities when $r = \tilde{\Theta}(k)$. More generally, it seems like the above approach should work only when r is significantly larger than k such as when $r = \Omega(k^2)$. On the other hand, our constructions in Section 4 enable low error probabilities for arbitrary values of r and k including the above setting when $r = \tilde{\Theta}(k)$.

We also note the single-server construction in [22] utilizes one of the above constructions [24,46,62] in a blackbox manner and, thus, suffers the same issues.