Limits of Preprocessing for Single-Server PIR*

Giuseppe Persiano† Kevin Yeo‡

Abstract

We present a lower bound for the *static* cryptographic data structure problem of *single-server private information retrieval (PIR)*. PIR considers the setting where a server holds a database of n entries and a client wishes to privately retrieve the i-th entry without revealing the index i to the server. In our work, we focus on *PIR with preprocessing* where an r-bit *hint* may be computed in a preprocessing stage and stored by the server to be used to perform private queries in expected time t. We consider the *public* preprocessing setting of Beimel et al. [JoC, 2004] where the hint is publicly available to everyone including the adversary.

We prove that for any single-server *computationally secure* PIR with preprocessing it must be that $tr = \Omega(n \log n)$ when $r = \Omega(\log n)$. If $r = O(\log n)$, we show that $t = \Omega(n)$. Our lower bound holds even when the scheme errs with probability $1/n^2$ and the adversary's distinguishing advantage is $1/n$. Our work improves upon the $tr = \Omega(n)$ lower bound of Beimel et al. [JoC, 2004].

We prove our lower bound in a variant of the *cell probe model* where only accesses to the memory are charged cost and computation and accesses to the hint are free. Our main technical contribution is a novel use of the *cell sampling technique* (also known as the incompressibility technique) used to obtain lower bounds on data structures. In previous works, this technique only leveraged the correctness guarantees to prove lower bounds even when used for cryptographic primitives. Our work combines the cell sampling technique with the privacy guarantees of PIR to construct a powerful, polynomial-time adversary that is critical to proving our higher lower bounds.

1 Introduction

Private information retrieval (PIR) is a static cryptographic data structure problem that considers the setting where one or more servers hold a database of n entries. Each of the n entries is uniquely indexed by an integer from $\{1, 2, \ldots, n\}$. A client wishes to retrieve the i-th entry from the database without revealing query index i to the subset of servers that may be corrupted by an adversary. The notion of PIR was first introduced in [17] for the setting of multiple servers with information-theoretic security. For multiple servers, it is naturally assumed that the adversary only corrupts a strict subset of the servers. See [3, 16, 5, 64, 28, 27] and references therein as examples of follow-up works. Kushilevitz and Ostrovsky [41] were the first to study the problem of PIR in the single-server setting with computational security where the server is compromised by the adversary with many follow-up works such as [13, 6, 24, 42, 33, 50].

One drawback of PIR is that the required computational cost per query is large thus prohibiting the usage of PIR in many large-scale applications. Intuitively, the server must operate on each of the n entries in a single-server PIR protocol. Otherwise, the server learns that the untouched entries were not queried by the user. Beimel, Ishai and Malkin [7] formalized this intuition and showed that linear server computation is required.

To overcome linear server computation, Beimel, Ishai and Malkin [7] presented the notion of a *PIR with preprocessing* where the server may compute a r-bit hint in a preprocessing stage and store the hint in addition to the original database. During query time, the hint may be used by the server to answer queries with, hopefully, sub-linear server computation t. For PIR with preprocessing, Beimel et al. [7] present a

*This is the revised version of [58].

†Università di Salerno. giuper@gmail.com

‡Google and Columbia University. kwlyeo@google.com

lower bound of $tr = \Omega(n)$ where t is the server computation time during queries and r is the size of the preprocessed hint. For the case of k servers, Beimel et al. [7] manage to obtain sublinear query times of $O(n/(\epsilon \log n)^{2k-2})$ by storing hints of size $O(n^{1+\epsilon})$. Clearly, there is a very wide gap between upper and lower bound and no progress has been made in the past two decades.

1.1 Our Contribution

This paper presents a higher lower bound for PIR with preprocessing in the single-server setting. To prove our lower bound, we utilize a very weak cost model where only memory accesses to the database are charged cost. All other operations may be performed free of charge, including computation, randomness generation, etc. Of particular note, memory accesses to the computed hint may be performed free of charge. Our model is a variant of the *standard PIR model* [7] and *cell probe model* [63]. Furthermore, our lower bounds hold for PIR schemes that may be adaptive and access data entries sequentially. For more on the lower bound model, we refer readers to Section 1.3. Specifically, we prove the following.

Theorem 1 (Informal)**.** *Consider any single-server, computationally-secure PIR with preprocessing scheme that errs with probability at most $1/n^2$ and the adversary's distinguishing advantage is at most $1/n$ for a database consisting of n entries. If the hint is at most r bits and queries probe at most t entries in expectation, it must be that $tr = \Omega(n \log n)$, when $3 \log n \leq r \leq n$. If $r < 3 \log n$, it must be that $t = \Omega(n)$.*

The above theorem states that for any hope to obtain sub-linear server computation, the hint must have size $\omega(\log n)$ bits. For any scheme with $O(\log n)$ bits, it must be that the server computation time remains linear and the standard single-server PIR algorithms remain optimal. Furthermore, we note that our $tr = \Omega(n \log n)$ is higher than the prior $tr = \Omega(n)$ lower bound [7] and matches the highest lower bounds known for non-cryptographic data structures in similar cost models (see Section 1.3 for more details). However, the gap between lower and upper bound for single-server PIR with preprocessing remains quite wide even after our work.

Unfortunately, we note that there are fundamental barriers to improving our lower bound. Prior works [7, 10] have discussed the connection between PIR with preprocessing with adaptive queries (as in our paper) and branching programs. Significantly improving our lower bound would directly imply stronger lower bounds for branching programs, which has been a long-standing open problem in the theoretical computer science community.

Our lower bound also separates PIR with preprocessing from other PIR models that also aim to obtain sub-linear server computation. For example, the above $tr = \Omega(n \log n)$ lower bound shows that the PIR with preprocessing model is harder than the offline-online PIR model where $tr = O(n)$ computationally-secure constructions are known [20]. Furthermore, our lower bounds also apply to public-key doubly efficient PIR. See Section 1.2 for discussions on related PIR models.

1.2 Related Works

The PIR with preprocessing model introduced by Beimel, Ishai and Malkin [7] is only one of several PIR models that have been studied to try and obtain sub-linear server computation. In this section, we outline other similar PIR models. Finally, we also overview prior works that proved lower bounds for cryptographic data structures.

Offline-Online PIR. Another model studied is the *offline-online PIR model*. During the *offline* phase, the client has access to the n entries and is allowed to *privately* construct a hint to be used for the upcoming future queries. The hint is computed without knowledge of the queries that will be arriving during the online phase and it is stored by the client privately from the adversary's view. In this setting, we may assume that the hint is no larger than the database without loss of generality (as a trivial solution is for the client to store the whole database as the private hint). When the client receives the query during the *online* phase, the client attempts to leverage the hint obtained during the offline phase to efficiently retrieve the desired entry. Several works have studied the offline-online PIR model such as [23, 7, 8, 52].

Corrigan-Gibbs and Kogan [20] were the first to present to present offline-online PIR schemes that had sub-linear server computation during query time. In particular, they present several $tr = O(n)$ constructions[1] in both the information-theoretic, two-server setting and computational, single-server setting. Furthermore, the authors present a lower bound of $tr = \Omega(n)$ showing that their above constructions were optimal.

The main difference between the offline-online PIR model and the PIR with preprocessing model is the hint. In offline-online PIR, the hint is computed by the client. Therefore, the hint is not a part of the adversary's view and cannot be used to construct attacks. For PIR with preprocessing, the hint is computed by the server, and, thus, may be used by the adversary when mounting attacks. In other words, the offline-online PIR model can be seen as PIR with *private* preprocessing.

Clearly, it seems that constructing PIR with preprocessing schemes should be significantly harder than devising offline-online PIR schemes as the adversary views the hint in one model, but not in the other. Our work is the first to formalize this separation by presenting higher lower bounds for PIR with preprocessing thus formally showing that it is a harder model than offline-online PIR.

Doubly-Efficient PIR. In another model, the notion of *doubly-efficient PIR* was considered that lies between the PIR with preprocessing and offline-online PIR models. As a reminder, PIR with preprocessing assumes the entire hint is seen by the adversary while offline-online PIR assumes the entire hint is not viewed by the adversary. For doubly-efficient PIR, the preprocessed hint is broken into two parts: the portion hidden from the adversary and the portion that is viewed by the adversary. The works of Boyle, Ishai, Pass and Wootters [10] and Canetti, Holmgren and Richelson [14] construct doubly-efficient PIR schemes where clients maintain a long-term private key and the server stores an encoding of the database that is generated using the private key. These constructions are able to obtain sub-linear server times in the single-server setting at the cost of $n^{O(1)}$ database encodings that are polynomial in the original database size. By using a new assumption based on permuted puzzles of Boyle, Holmgren and Weiss [9], both constructions may be shown to be secure for an unbounded number of queries.

Using obfuscation, Boyle, Ishai, Pass and Wootters [10] also present constructions that forego the need for the client private key and enable the hint to be entirely public. They denoted this primitive as *public-key doubly-efficient PIR*. We note that the main difference between public-key doubly-efficient PIR and PIR with preprocessing is that public-key doubly-efficient PIR enables randomization in the hint generation and restricts the querier to be non-adaptive. All the lower bounds in our paper also apply to public-key doubly-efficient PIR as we also consider randomized hint generation and general adaptive query algorithms.

Hamlin, Ostrovsky, Weiss and Wichs [36] present an extension of doubly-efficient PIRs, denoted private anonymous data access (PANDA), where the client holds a private key and the server holds an encoding of the database based on the private key. The key difference is that PANDA requires the server to change the database's encoding after queries. The constructions of PANDA are able to achieve sub-linear (or even poly-logarithmic) server times during queries but still require $O(n^{1+\epsilon})$ server storage. Furthermore, both the server storage and time grow in the number of malicious clients (unlike the prior models).

Lower Bounds for Cryptographic Data Structures. The first cell probe lower bound for cryptographic data structures was presented by Larsen and Nielsen [46] that presented a tight lower bound for oblivious RAMs (ORAMs). This paper led to a line of works proving lower bounds for a variety of different settings and primitives including other oblivious data structures [39], differential private guarantees [56], weaker adversaries that do not see query boundaries [38], non-colluding multiple-server settings [47], oblivious near-neighbor search [45], encrypted search [54] and smaller block sizes [40]. We note that all these lower bounds are for dynamic data structures and utilize either the information transfer [55] or chronogram techniques [29]. None of the lower bound techniques that are used in our paper resemble the ones used in these prior papers.

Several works have also shown the difficulty of proving lower bounds for cryptographic data structures. Boyle and Naor [11] showed that lower bounds for offline ORAMs would imply non-trivial circuit lower bounds. Weiss and Wichs [62] proved that lower bounds for online, read-only ORAMs would result in

[1] We note that the constructions in [20] were not tight with respect to hint size and server time as the authors focused on reducing communication in some settings. We note it is straightforward to construct tight constructions using their techniques. We sketch these constructions in Appendix A.

proving either proving non-trivial circuit or locally decodable code lower bound. Both of these are long-standing problems in theoretical computer science.

We note that other works have considered more restrictive models such as the "balls-and-bins" model. In this model, there have been lower bounds for oblivious RAMs [34] as well as one-round oblivious RAMs [15].

1.3 The Lower Bound Model

The cell probe model [63] is a standard, strong model for proving data structure lower bounds. In this model, all computation is free. The only charged costs are the size of the data structure and the number of memory cells that are accessed during queries. As the model is weak, cell probe lower bounds apply to almost any imaginable computational model including the standard RAM model. Furthermore, the lower bound is independent of other variables such as CPU power. Unfortunately, proving cell probe lower bounds ends up being difficult due to the weak cost model.

PIR is typically studied as a static data structure that only supports read queries and no write queries. The static nature of the data structure adds to the technical difficulty of our results. The current highest lower bounds in the cell probe model for static data structures are $\tilde{\Omega}(\log n)$ [44, 35] for databases of size n. Furthermore, there is evidence that proving higher static lower bounds will be difficult [26, 61]. Even for dynamic data structures, the best lower bounds are only logarithmically higher at $\tilde{\Omega}(\log^2 n)$ [43] that also match the highest known lower bounds for cryptographic data structures [45]. Therefore, cell probe lower bound techniques for unrestricted data structures seem very far from proving meaningful PIR lower bounds.

Proving lower bounds for read-only data structures has also proven difficult. Consider oblivious RAMs (ORAMs). For dynamic ORAMs with write operations, tight $\Omega(\log n)$ lower bounds [46] are known. For read-only ORAMs, a recent work by Weiss and Wichs [62] shows that proving non-trivial lower bounds would imply new lower bounds for either sorting circuits or locally decodable codes (both are long-standing open problems).

Due to this difficulty, prior works in cryptography have proved lower bounds in models that are more restrictive than the cell probe model. Some examples include the "balls-and-bins" model [34, 11] with a non-coding assumption for various cryptographic data structures [1, 53, 15]. Similar restricted models have been used outside of cryptography such as proving lower bounds for non-adaptive data structures [12, 59, 35, 57]. In our work, we will also consider a slightly more restrictive model to prove our higher lower bounds.

Standard PIR with Preprocessing Model. We will prove our lower bounds in a variant of the *standard PIR model* as defined by Beimel *et al.* [7] where the server only stores a copy of the database. Most prior PIR schemes such as [17, 41] are either designed in the standard PIR model, or may be converted to the standard PIR model with no additional asymptotic overhead.

We adapt the standard PIR model slightly for preprocessing where the hint is stored by the server in addition to the original database. We denote this as the *standard PIR with preprocessing model*. The standard PIR model ends up being a very good model as almost all prior constructions of PIR with preprocessing may be constructed in the standard PIR model with no additional asymptotic costs. To our knowledge, all known PIR with preprocessing schemes [7] as well as similar models [10, 14, 52, 36, 20, 60] were either constructed directly in the standard PIR model or may be converted to the standard PIR model without any additional asymptotic overhead.

The usage of the standard PIR with preprocessing model is not new for lower bounds for PIR. The prior $tr = \Omega(n)$ lower bounds for PIR with preprocessing [7] and offline-online PIR [20] were both proven in a similar model.

Relation to Succinct Cell Probe Model. We discuss interesting relationships between the standard PIR with preprocessing model and other lower bound models. First, we will show that the standard PIR with preprocessing model is essentially equivalent to *systematic succinct cell probe model*. Systematic succinct data structures force the designer to store the database in read-only memory without encoding. Additionally, the data structure is given a small portion of r-bit memory that may be modified. We note that there is a direct equivalence to the standard PIR with preprocessing model where the database must be stored without encoding and the designer is given an r-bit preprocessed hint that may be modified over time. Several prior

works have proven lower bounds for systematic succinct data structures such as [22, 30, 37] with the highest lower bounds peaking at $tr = \Omega(n \log n)$. We note that our lower bound for computationally-secure PIR with preprocessing matches the highest lower bounds known in the systematic succinct cell probe model.

Relation to Oracle Models. In a line of work closer to cryptography, we note that the standard PIR with preprocessing model is similar to many oracle models that have been considered in the past. At a high level, there is some task that must be accomplished by the algorithm designer that only has access to some oracle f that when given x returns $f(x)$. Additionally, the algorithm is able to preprocess some r-bit hint about the oracle f before receiving the task. When attempting to finish the task, the goal of the algorithm is to reduce the number of queries sent to the oracle. Once again, this is highly similar to the standard PIR model. We can view the database as an oracle f that returns the i-th database entry when given input i. In both models, the algorithm designer is able to preprocess some r-bit hint that may be used during online queries. Many prior works (such as [32, 31, 4, 21, 2, 25, 18, 19] and references therein) have proven lower bounds in these oracle models for many tasks such as function inversion and subverting random oracles.

2 Technical Overview

In this section, we present a technical overview for our main result: the $tr = \Omega(n \log n)$ lower bound for computationally-secure PIR with preprocessing schemes. At a high level, our proof may be broken down into four main parts:

1. Constructing the hard distribution.
2. Upper bounding the number of probed non-zero entries in the database.
3. Using privacy guarantees to construct a powerful, PPT adversary.
4. Completing the proof using a final impossible encoding.

We outline each of the four parts below.

Hard Distribution. The main inspiration of our lower bound comes from the following observation. Consider any PIR with preprocessing scheme that uses r-bit hints. Suppose the database of n entries was partitioned into r parts each of n/r entries. Within each part, we embed a non-zero entry at a uniformly random location amongst the n/r possible entries. Without any preprocessing or hints available, a PIR scheme would be expected to access $\Omega(n/r)$ entries within the part to find the location of the non-zero entry. Let us consider how an r-bit hint may be useful to speed up the algorithm to find the non-zero entry within each part. The simplest way is to encode the location of the non-zero entry in each part using $\log(n/r)$ bits. Unfortunately, there is not enough space in the r-bit hint. In particular, the r-bit hint may encode one bit of information on average for each of the r parts. With only one bit for each part, one may only reduce the possible search space within each part to $n/(2r)$ possible locations for the non-zero entry still requiring accessing $\Omega(n/r)$ entries on average to find the non-zero entry within each part. Even with an r-bit hint, it seems that the problem of finding all non-zero entry locations in each of the r parts is almost as hard as the original problem without the r-bit hint. With the above intuition, we construct our hard database distribution $\mathbf{D}$ as embedding a random non-zero entry at a random location within each of the r parts.

Bounding Number of Probed Non-Zero Entries. To formalize the above hardness, we start by proving a result about the number of non-zero entries that may be probed during queries. For a contradiction, suppose that there exists a PIR with preprocessing scheme with r-bit hints and $tr = o(n \log n)$. This means $t = o(n \log n/r)$. Since finding a non-zero entry within any part seems to take $\Omega(n/r)$ probes, the scheme should be able to only find $o(\log n)$ non-zero entries per query. Unfortunately, this is not entirely correct as we assume that exactly one bit of information is encoded for each of the r parts. A scheme could instead encode the exact location of the non-zero entry for a small number of parts. A single non-zero location may be encoded using $\log(n/r)$ bits. As a result, one query could discover $r/\log(n/r)$ non-zero entries using the

hint. Over multiple queries, the hint will only aid queries to discover the same non-zero entries repeatedly. Therefore, it should be true that each query cannot discover more than $o(\log n)$ new non-zero entries on average over a large number of queries.

Formally, we present the notion of *maximally discovering* query sequences where each query has high probability to discover at least $1/16 \cdot \log n$ new non-zero entries that were not probed by prior queries. Furthermore, all queries performed after a maximally discovering query sequence will probe less than $1/16 \cdot \log n$ new non-zero entries with high probability. In other words, the sequence is maximal as one cannot append any more queries that discover a large number of non-zero entries. For most databases in the hard distribution, we show that any maximally discovering query sequence for the database must contain at most $r/\log n$ queries. Additionally, we show that the total number of non-zero entries probed by all $r/\log n$ queries in any maximally discovering query sequence cannot be more than $r/128$ with high probability. To prove this, we show that if either (or both) of the two statements are false, there exists an impossible one-way encoding of the database that contradicts Shannon's source coding theorem. At a high level, the encoder (Alice) may compress the non-zero entry locations beyond the information-theoretic lower bound by instructing the decoder (Bob) to observe the probed entries for a maximally discovering query sequence of at least $r/\log n$ queries while encoding only the subset of probed non-zero entries.

Using Privacy Guarantees. Our next technique is inspired by the incompressibility technique [32, 31, 21, 35] in cryptography and the cell sampling technique [51, 44] in data structures. At a high level, prior works took an impossibly efficient data structure and showed that a small subset of cells may be sampled that may be used to retrieve the answers for many queries. This is used to derive a contradiction that the subset cannot contain enough information to answer the queries. Instead, we utilize this idea to construct a PPT adversary that enforces very strict requirements about any scheme beating our lower bound. To our knowledge, this is the first time that the cell sampling (and the incompressibility technique) has been directly combined with privacy to construct a strong adversary to prove lower bounds.

Consider any PIR with preprocessing scheme with $tr = o(n \log n)$ that beats our lower bound. Suppose we execute any maximally discovering query sequence and record all probed non-zero entries. Any later query will probe at most $\log n/16$ non-zero entries with high probability. Consider the experiment where a subset $\mathbf{T}$ of non-zero entries is sampled such that each non-zero entry of the database appears in $\mathbf{T}$ with some constant independent probability such as $1/10$. For any query that probes at most $\log n/16$ new non-zero entries, the probability that all the probed non-zero entries were also sampled in $\mathbf{T}$ is $(1/10)^{\log n/16} = 1/\mathsf{poly}(n)$. When this is true, we denote that the query is *resolved* by the subset $\mathbf{T}$. The key takeaway is that for any query q that probes at most $\log n/16$ new non-zero entries with high probability, it is possible for a PPT adversary to find a small subset $\mathbf{T}$ containing only $(1/10)$-fraction of all non-zero entries such that the query q is resolved by $\mathbf{T}$ with $1/\mathsf{poly}(n)$ probability. This immediately implies that for any other query $q' \neq q$, a similar statement should hold as, otherwise, the PPT adversary may compromise privacy by observing whether the challenge query is resolved by $\mathbf{T}$ or not. In other words, the same subset $\mathbf{T}$ must also resolve any other query $q' \neq q$ with probability not too much smaller than the probability that $\mathbf{T}$ resolves q to ensure that a PPT adversary cannot distinguish between queries to the q-th and q'-th entry.

Final Impossible Encoding. In the final step, we utilize the strong property derived using the privacy guarantees of PIR to present an impossible final encoding to complete the proof. At first, this could seem similar to the final step of prior works using the cell sampling/incompressibility technique [32, 31, 51, 21, 44, 35] where the small subset $\mathbf{T}$ is too small to store all the information contained in the query's outputs. Unfortunately, there is a subtle but significant difficulty since we only sample amongst non-zero entries. It is not sufficient for a decoder (Bob) to only know the contents and locations of the subset $\mathbf{T}$. In particular, Bob cannot distinguish between zero entries and non-zero entries that were not sampled into $\mathbf{T}$. As a result, Bob cannot determine whether a query is resolved. A similar problem appears when proving super-logarithmic dynamic data structure lower bounds (see [43, 44, 48, 45] as some examples).

To get around this problem, we show that it suffices for the encoder (Alice) to also send the locations, but not the contents, of all non-zero entries that are not sampled into $\mathbf{T}$. In more detail, Alice will send Bob the subset of $\mathbf{T}$ of non-zero entries (both locations and contents) that resolve all queries $q \in [n]$ with non-zero probability as well as the locations of non-zero entry locations that do not appear in $\mathbf{T}$. To decode, Bob

will execute all queries $q \in [n]$ with all possible random strings until finding a random string that ensures q is resolved by the encoded subset $\mathbf{T}$. We emphasize that Bob needs to know the location of all non-zero entry locations to determine whether a query q is resolved by $\mathbf{T}$ or not. Consider an execution of q with a random string that probes an entry outside of $\mathbf{T}$. If Bob did not know the location of all non-zero locations, then Bob will be unable to determine whether any probed entry not in $\mathbf{T}$ is actually a zero entry (for which Bob knows the contents) or a non-zero entry outside of $\mathbf{T}$ (for which Bob does know the contents). By finding a random string such that the execution of q is resolved by $\mathbf{T}$, Bob will successfully execute the query q correctly and produce the q-th entry. After executing all queries $q \in [n]$, Bob successfully decodes the database. This contradicts Shannon's source coding theorem as Alice only encodes the contents of $r/10$ non-zero entries while Bob retrieves the contents of all r non-zero entries.

The above description only considers the setting where the PIR with preprocessing scheme always produces the correct answer. In our proof, we prove a lower bound for schemes that err with probability at most $1/n^2$. To incorporate error, we prove that strictly more than half the random strings that result in a query q being resolved by subset $\mathbf{T}$ will also produce the correct answer. Therefore, the above decoding algorithm may be modified such that Bob executes q with all possible random strings. For all random strings that result in q being resolved by $\mathbf{T}$, Bob takes the majority answer that will be guaranteed to be the correct contents of the q-th entry.

3 Definitions

3.1 PIR with Preprocessing

We start by defining the notion of a PIR with preprocessing and then define the notions of error and security for which we are going to prove out lower bound.

Definition 1 (PIR with Preprocessing)**.** *A* PIR with preprocessing *scheme is a pair of randomized, efficient algorithms* $\Pi = (\mathsf{Preprocess}, \mathsf{Query})$ *such that*

1. $H \leftarrow \mathsf{Preprocess}(D, R_H)$*: The* preprocess *algorithm takes the database* D *and a random string* R_H *to compute the preprocessed* r*-bit hint* H*.*

2. $\mathsf{res} \leftarrow \mathsf{Query}(q, H, R; D)$*: The* query *algorithm is jointly executed by the client and server. The client receives as input the query index* $q \in [n]$*, the hint* H *and coin tosses* R *while the server receives as input the database* D *of length* n*. Once the query algorithm is completed the client receives* res*, algorithm's attempted response to query* q*. The algorithm may be adaptive and use multiple rounds between the client and server.*

In the original PIR with preprocessing definition [7], the hint generation was assumed to be deterministic. However, the definition of public-key doubly-efficient PIR [10] extended the notion to enable randomized hint generation. In our definition, we enable random hint generation to prove lower bounds in both models. Additionally, we enable the query algorithm to be adaptive. As a result, our lower bounds apply to both adaptive (as defined for PIR with preprocessing) and non-adaptive query algorithms (as defined for public-key doubly-efficient PIR).

We next define the error probability for a PIR with preprocessing.

Definition 2 (Correctness)**.** *We say that PIR with preprocessing* Π *errs with probability at most* ϵ *if, for every database* D *and every query* q*, it holds that*

$$\Pr_{\mathbf{R}_H, \mathbf{R}}[\mathbf{Q}(D, q) \neq D_q] \leq \epsilon$$

where

$$\mathbf{Q}(D, q) := \left\{ \mathsf{res} : \begin{array}{l} H \leftarrow \mathsf{Preprocess}(D, \mathbf{R}_H); \\ \mathsf{res} \leftarrow \mathsf{Query}(q, H, \mathbf{R}; D) \end{array} \right\}.$$

To formally define our security notion we consider, for $\eta = 0, 1$, the following game $\mathsf{IndGame}^{\eta}_{\mathcal{A}}$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$.

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

1. The challenger $\mathcal{C}$ runs $H \leftarrow \mathsf{Preprocess}(D, R_H)$ on randomly chosen coin tosses R_H.
2. The adversary $\mathcal{A}(D, H)$ on input D and H outputs two queries q^0 and q^1.
3. The challenger $\mathcal{C}$ executes query q^η using the hint H and randomly chosen coin tosses R. The challenger records the *transcript* $\mathcal{T}$ of the probes to D performed by the Query algorithm.
4. The challenger returns the transcript $\mathcal{T}$ to $\mathcal{A}$.
5. $\mathcal{A}(\mathcal{T})$ outputs a bit b.

We let $p^{\eta}_{\mathcal{A}}(D)$ denote the probability that $\mathcal{A}$ outputs 1 in game $\mathsf{IndGame}^{\eta}_{\mathcal{A}}(D)$. The probability is taken over the randomness of the adversary $\mathcal{A}$ and of the challenger $\mathcal{C}$.

Definition 3 (Security)**.** *A* PIR with preprocessing *is computationally δ-secure if for all probabilistically polynomial time (PPT) adversaries $\mathcal{A}$ and all sufficiently large databases D, the following holds:*

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

We note the above may be modified for information-theoretically secure PIR with preprocessing schemes by considering all adversaries $\mathcal{A}$ without any bounds on computation time.

Definition 4 ((ϵ, δ)-Secure)**.** *We define a PIR with preprocessing Π to be (ϵ, δ)-secure if it has error probability at most ϵ and is δ-secure.*

Definition 5 ((r, t)-Efficient)**.** *We define a PIR with preprocessing Π to be (r, t)-efficient if the following holds:*

1. *For all possible databases D and all possible coin tosses R_H, the hint H has length at most r.*
2. *For all possible databases D and all queries q, the expected number of entries of D probed when executing query q on database D is at most t over the random choices of R_H and R.*

3.2 Lower Bounds in the Standard PIR with Preprocessing Model

We formalize the model that we will use to prove our lower bounds. The cell probe model was introduced by Yao [63] to prove data structure lower bounds. Larsen and Nielsen [46] (see also [39]) adopted the notion for cryptographic data structures by introducing the oblivious cell probe model. In this model, there exist two parties: a client and a server. The server's memory consists of w-bit cells where each cell has a unique integer address. Finally, there exists a long, uniformly random binary string $\mathbf{R}$. The string $\mathbf{R}$ is able to be utilized as the random coin tosses for randomized algorithms and it is drawn uniformly at random before any operations of the data structure are performed. Note that $\mathbf{R}$ may be used as a random oracle. We will assume that the random string $\mathbf{R}$ may be arbitrarily long, but finite. At first, this assumption may seem to rule out using algorithms that may use infinitely long randomness with error probabilities decreasing with the randomness length and/or running time. Instead, we can consider a variant of the algorithm that uses arbitrary long, but finite, randomness but provides an answer once the error probability is small enough. As we prove lower bounds for algorithms that potentially err, our results also hold for such algorithms.

In our model, a probe refers to either reading or writing to a single server memory cell. The running time of a data structure is the number of cell probes that are performed when performing a query. Note, this is the only cost incurred by the data structure. The data structure is able to access the random string $\mathbf{R}$ and hint free of charge. Furthermore, the data structure is able to perform arbitrarily long computation without any cost.

Throughout our paper, we will adapt the oblivious cell probe model to be compatible with the standard PIR model that also enables preprocessing. In the standard PIR model, the database D consisting of n entries each of b bits is stored in plaintext. With respect to the cell probe model, we will assume that D is stored on the server using n cells each consisting of $w = b$ bits. Therefore, the total time for any data structure t will correspond to the expected number of entries of D that are accessed during a query. For our model, we will assume accessing the hint is free of charge. The only cost during query time is the number of entries probed in D.

Finally, we note that there is an additional preprocessing algorithm utilized by to construct the hint H. In our model, we will not charge any cost to the preprocessing model in terms of computation even if it requires probing entries in D. The only limitation is that the hint must be at most r bits in length.

As a remark, we note that our lower bound model is very weak. For example, we do not change any cost to computation, randomness generation, preprocessing to create the hint H, etc. By proving a lower bound in a weaker model, our results are stronger as they also apply for any reasonable model where all of the above are charged reasonably.

4 A Lower Bound for Computational PIR with Preprocessing

In this section, we will prove the following lower bound on PIR with preprocessing schemes that are secure against computationally-bounded adversaries.

Theorem 2. *Let* $\Pi = (\mathsf{Preprocess}, \mathsf{Query})$ *an* (ϵ, δ)*-secure,* (r, t)*-efficient PIR with preprocessing scheme with* $\epsilon \leq 1/n^2$ *and* $\delta \leq 1/n$*. For a database with* n *entries, then* $tr = \Omega(n \log n)$ *for* $3 \log n \leq r \leq n/4$*. If* $r < 3 \log n$*, then* $t = \Omega(n)$*.*

We prove that above for database with n b-bit entries and $b \geq 4$. This suffices also for single-bit ($b = 1$) entries, as any PIR with preprocessing scheme for a 1-bit entry database may be used to construct a scheme for a 4-bit entry database by repeating the algorithm four times. Furthermore, this reduction incurs no additional asymptotic overhead.

Our choices of $\epsilon \leq 1/n^2$ and $\delta \leq 1/n$ are for convenience. By carefully choosing constants in our proof, one can redo the proof for $\epsilon = 1/n^{c_1}$ and $\delta = 1/n^{c_2}$ for constants $0 < c_1 < 2$ and $0 < c_2 < 1$. We note our lower bound immediately holds for typical negligible distinguishing advantages $\delta = \mathsf{negl}(n)$.

In our proof, we will assume that $r \geq 3 \log n$. For the case, when $r < 3 \log n$, we will arbitrarily pad the hint until it contains at least $3 \log n$ bits. Even with this padding, note that a lower bound of $tr = \Omega(n \log n)$ implies that $t = \Omega(n)$ when $r < 3 \log n$. The upper bound of r enforces that the hint can store at most 1/4-fraction of the database as $b \geq 4$. In other words, our lower bound holds even when the hint is large enough to store a constant fraction of the database. Note, there are trivial algorithms in our model when the hint is very large that rule out higher lower bounds. For example, if $r = b \cdot (n - 1)$, the hint can contain $n - 1$ database entries and queries would only need to probe one database entry as accessing the hint is free.

4.1 Hard Distribution

To prove our theorem, we start by defining the hard distribution $\mathbf{D}$ over databases of n b-bit entries, with $b \geq 4$. We consider the n entries as partitioned into exactly r *parts* where each part consists of exactly n/r consecutive entries, where r is the size of the hints in bits. To sample a database D according to $\mathbf{D}$, we choose, for each of the r parts, exactly one entry uniformly at random amongst all n/r possible locations to be a non-zero string. The contents of the non-zero entry is chosen uniformly chosen from among all b-bit string that start with a 1-bit. All the other $n/r - 1$ entries of the part are set equal to 0^b (that is, to a string of b 0's). The next lemma gives the entropy of the hard distribution $\mathbf{D}$.

Lemma 1. $H(\mathbf{D}) = r\big(\log(n/r) + (b - 1)\big)$.

Proof. $\mathbf{D}$ is the product of r identical independent distributions and thus we can simply focus on each smaller distribution that picks a single non-zero entry from n/r possible locations. The location has $\log(n/r)$ bits

of entropy. The non-zero entry is uniformly selected from a set of 2^{b-1} strings resulting in $(b-1)$ bits of entropy. Therefore, each distribution has $\log(n/r) + (b-1)$ bits of entropy. □

4.2 Bounding Number of Probed Non-Zero Entries

The first part of our proof looks at the non-zero entries of the databases of the hard distribution that are probed by queries. We start by proving that the number of probed non-zero entries cannot be too large for any PIR with private preprocessing $\Pi = (\mathsf{Preprocess}, \mathsf{Query})$ for which $tr = o(n \log n)$.

Formally, we show that sequences of not "too many" queries cannot discover "too many" non-zero entries of D. We will introduce notation to make precise statements.

Notation. For a sequence $S = (s_1, \ldots, s_l)$ we denote by $S_{<i}$ the sequence $(s_1, \ldots, s_{i-1})$ of the first $i-1$ components of S and by $S_{\leq i}$ the sequence $(s_1, \ldots, s_i)$ of the first i components of S. For a database D, we denote $\mathsf{nz}(D)$ as the set $\mathsf{nz}(D) = \{i \in [n] \mid D_i \neq 0^b\}$ of the indices of the non-zero entries of database D.

For any query $q \in [n]$, we denote $\mathsf{nzP}(q, D, H, R) \subseteq \mathsf{nz}(D)$ as the set of the indices of non-zero entries of D probed by algorithm $\mathsf{Query}(q, H, R; D)$ run for query q, using hint H and coin tosses R on database D. We extend this notation to a sequence $Q = (q_1, \ldots, q_k)$ of queries and we denote by $\mathsf{nzP}(Q, D, H, R)$ as the set of indices of non-zero entries of D probed by executing of the queries in Q using H as the hint for all queries. That is,

$$\mathsf{nzP}(Q, D, H, R) := \mathsf{nzP}(q_1, D, H, R_1) \cup \ldots \cup \mathsf{nzP}(q_k, D, H, R_k).$$

For every query q, sequence Q of k queries, database D, hint H, and coin tosses $R = (R_1, \ldots, R_k, R_{k+1})$, we define the *contribution* $\mathsf{contrib}(q, Q, D, H, R)$ of query q w.r.t. Q as

$$\mathsf{contrib}(q, Q, D, H, R) := \mathsf{nzP}(q, D, H, R_{k+1}) \setminus \mathsf{nzP}(Q, D, H, R_{\leq k}).$$

In other words, the *contribution* of query q for sequence Q is the set of non-zero cells that are probed if q is executed after Q is completed and are not probed during the execution of the first k queries.

When we consider random hints $\mathbf{H}$, we will also consider the randomness $\mathbf{R}_H$ that is the randomness used by the $\mathsf{Preprocess}$ algorithm to generate the random hint. Typically, we use $\mathbf{H}$ as shorthand for $\mathsf{Preprocess}(D, \mathbf{R}_H)$.

Next, we define the notion of a *maximally discovering* query sequence. These sequences ensure that each query in the sequence has large contribution (in terms of the number of probed non-zero indices that were previously undiscovered). Furthermore, they are maximal in the sense that it is not possible to append any query with large contribution.

Definition 6. *For any database D and query sequence Q, we say that Q is* maximally discovering *for database D if the following properties hold:*

1. *For every query q_i in $Q = (q_1, \ldots, q_k)$, it holds that*

$$\Pr_{\mathbf{R}_H, \mathbf{R}} \left[\left| \mathsf{contrib}(q_i, Q_{<i}, D, \mathbf{H}, \mathbf{R}) \right| \geq \frac{\log n}{16} \right] \geq \frac{1}{4}.$$

2. *For every $i = 1, \ldots, n$, it holds that*

$$\Pr_{\mathbf{R}_H, \mathbf{R}} \left[\left| \mathsf{contrib}(i, Q, D, \mathbf{H}, \mathbf{R}) \right| < \frac{\log n}{16} \right] > \frac{3}{4}.$$

At a high level, every query in a maximally discovering sequence has contribution at least $\log n/16$. Furthermore, there is no query that may be appended to the sequence that has contribution $\log n/16$ with probability at least $1/4$.

We are now ready to define the notion of a *typical* database.

Definition 7. *We say that a database D of size n is* typical *in symbols $\mathtt{E}^{\mathtt{D}}(D) = 1$, if for any maximally discovering query sequence Q, the following properties hold:*

1. $|Q| \leq r/\log n$.

2. $\Pr_{\mathbf{R}_H,\mathbf{R}}\left[\left|\mathsf{nzP}(Q, D, \mathbf{H}, \mathbf{R})\right| < \frac{r}{128}\right] \geq \frac{31}{32}$.

Otherwise, we say that D is non-typical *and write* $\mathtt{E}^{\mathtt{D}}(D) = 0$.

In other words, if a database D is typical, then we expect that all maximally discovering query sequences for D are not too long. Furthermore, we expect these maximally discovering sequences to be unable to find too many unique non-zero entries (less than $r/128$) most of the time.

If a database D is non-typical, then there exists a maximally discovering query sequence Q that may be used to compress D very well. In more detail, the maximally discovering query sequence reveals many non-zero entries of D. We should not expect most databases in the support of the hard distribution $\mathbf{D}$ to be non-typical. Otherwise, we will be able to impossibly compress databases drawn from $\mathbf{D}$ using fewer than $H(\mathbf{D})$ bits in expectation as we show later.

Before making the above statement formal, we first formalize the argument that non-typical databases D enable discovery of many non-zero entries. We prove the following technical lemma that says that if a database is not typical, i.e., $\mathtt{E}^{\mathtt{D}}(D) = 0$, then there exists a maximally discovering sequence Q which allows the discovery of a *large* number of non-zero entries of D with some *constant* probability. We will then use this fact to succinctly encode D.

Lemma 2. *If $\mathtt{E}^{\mathtt{D}}(D) = 0$, then there exists a maximally discovering query sequence Q such that*

$$\Pr_{\mathbf{R}_H,\mathbf{R}}\left[\left|\mathsf{nzP}(Q, D, \mathbf{H}, \mathbf{R})\right| \geq \frac{r}{128}\right] \geq \frac{1}{32}.$$

Furthermore, if $|Q| > r/\log n$, then

$$\Pr_{\mathbf{R}_H,\mathbf{R}}\left[\left|\mathsf{nzP}(Q_{\leq r/\log n}, D, \mathbf{H}, \mathbf{R})\right| \geq \frac{r}{128}\right] \geq \frac{1}{32}.$$

Proof. If D violates the second condition of the Definition 7, then the lemma follows directly. Suppose D only violates the first condition and $|Q| = k > r/\log n$. By Definition 6, we know that for each query q_i of a maximally discovering sequence,

$$\Pr_{\mathbf{R}_H,\mathbf{R}}\left[\left|\mathsf{contrib}(q_i, Q_{<i}, D, \mathbf{H}, \mathbf{R})\right| < \frac{\log n}{16}\right] \leq \frac{1}{4}.$$

Consider the random variable X of the number of indices i where $\left|\mathsf{contrib}(q_i, Q_{<i}, D, \mathbf{H}, \mathbf{R}_{\leq i})\right| < \log n/16$ in the first $k = r/\log n$ queries of Q. The random variable X is taken over the random variables $(\mathbf{R}_H, \mathbf{R}_{\leq i})$ of the coin tosses used to generate the hint and perform the Query algorithm for the first i queries of Q. Then $\mathsf{E}[X] \leq 3k/4$ and thus, by Markov inequality, $\mathsf{Pr}[X \geq 7k/8] \leq 6/7$. This implies that with probability at least $1/7 \geq 1/32$, there are at least $k/8$ indices each contributing at least $\log n/16$ new non-zero cells. This means the total contribution is at least $(\log n/16) \cdot k > r/128$ completing the proof. □

We are now ready to show that if the scheme Π beats our lower bound then the hard distribution does not contain many non-typical databases.

Theorem 3. *Let $\Pi = (\mathsf{Preprocess}, \mathsf{Query})$ be a (r, t)-efficient PIR with preprocessing such that $tr \leq cn \log n$, for some sufficiently small constant $c > 0$. Then,*

$$\Pr_{\mathbf{D}}[\mathtt{E}^{\mathtt{D}}(\mathbf{D}) = 1] \geq 1/2$$

where $\mathbf{D}$ is the hard distribution.

Proof. For the sake of contradiction, suppose that

$$\Pr_{\mathbf{D}}[\mathtt{E}^{\mathtt{D}}(\mathbf{D}) = 0] > 1/2.$$

Consider the following one-way encoding problem in which Alice and Bob receive a sequence $R^\star = (R_1^\star, \ldots, R_k^\star)$ of k random and independently distributed strings as input. In addition, Alice gets database D as input and her task is to produce an encoding of D so that Bob can recovery D from the encoding and the sequence $R^\star$. Note that both Alice and Bob are computationally unbounded but communication is one-way from Alice to Bob. We will describe an encoding scheme that ensures that, when D is drawn according to the hard distribution $\mathbf{D}$, Alice's expected message size is smaller than the entropy of $\mathbf{D}$ given $R^\star$, thus obtaining a contradiction with Shannon's source coding theorem.

Alice's Encoding. Alice receives as input the database D and uniformly random strings $R^\star = (R_H^\star, R_1^\star, \ldots, R_k^\star)$ where $k = r/\log n$.

1. Alice checks whether $\mathtt{E}^{\mathtt{D}}(D) = 1$ or $\mathtt{E}^{\mathtt{D}}(D) = 0$. To do so, Alice iterates through all possible coin tosses for the $\mathsf{Preprocess}$ and Query algorithm.
2. If $\mathtt{E}^{\mathtt{D}}(D) = 0$,
   (a) Alice finds a query sequence Q that is a counterexample to $\mathtt{E}^{\mathtt{D}}(D) = 1$. If $|Q| \geq k = r/\log n$, then set $Q := Q_{\leq k}$.
   (b) Alice executes query sequence Q using $R^\star$ and counts the total number ℓ of different entries of D probed and the total number $\ell_{\mathtt{nz}} = \mathsf{nzP}(Q, D, H, R^\star)$ of different non-zero entries of D probed while executing the queries.
3. If $\mathtt{E}^{\mathtt{D}}(D) = 1$ or $\ell > 128tk$ or $\ell_{\mathtt{nz}} < r/128$, then Alice will encode D trivially as follows:
   (a) Alice starts the encoding with a 1-bit.
   (b) For each of the r non-zero entries of D, Alice uses $\log(n/r)$ bits to encode the index within the part and $(b-1)$ bits to encode the content.
4. If $\mathtt{E}^{\mathtt{D}}(D) = 0$ and $\ell \leq 128tk$ and $\ell_{\mathtt{nz}} \geq r/128$, then Alice encodes as follows:
   (a) Alice starts the encoding with a 0-bit.
   (b) Alice encodes the hint $H = \mathsf{Preprocess}(D, R_H^*)$ using r bits.
   (c) Afterwards, Alice encodes ℓ using $\log n$ bits. As ℓ denotes the number of unique entries accessed in D, we know that $0 \leq \ell \leq n$ and thus $\log n$ bits suffice for encoding ℓ.
   (d) Alice constructs the sequence $S = (S_1, \ldots, S_\ell)$ of the ℓ entries in D in the order they were first probed. Thus, S_i is the i-th unique entry probed when performing the queries in sequence Q. T is the subset $T \subseteq [\ell]$ such that $T := \{i \in [\ell] \mid D_{S_i} \neq 0^b\}$. Note that $|T| = \ell_{\mathtt{nz}} \geq r/128$.
   (e) Alice encodes the content of the first $r/128$ entries in T by using at most $\log \binom{\ell}{r/128}$ bits for the indices in D and $r/128 \cdot (b-1)$ bits for the contents.
   (f) Finally, Alice encodes the remaining $127/128 \cdot r$ non-zero entries of D. To do this, Alice iterates through the r parts of D in some fixed order. For each part whose non-zero entry has not been encoded in the previous step, Alice encodes the non-zero entry using $\log(n/r)$ bits for the location and $(b-1)$ bits for the contents.

Bob's Decoding. Bob receives as input the encoding from Alice. Additionally, Bob receives the same uniformly random strings $R_1^\star, \ldots, R_k^\star$ as Alice.

1. If Alice's encoding starts with a 1, then Bob decodes the location and contents of each non-zero entry using $\log(n/r) + b - 1$ bits. In this case, Bob successfully decodes D.

2. We now consider the case where Alice's encoding starts with a 0.

   (a) Bob decodes the hint H.

   (b) Bob decodes ℓ, the number of unique entries that will be probed when executing the queries in Q.

   (c) Bob also decodes the $r/128$ indices in the subset $T \subseteq [\ell]$. Recall that if $i \in T$, then $D_{S[i]} \neq 0^b$ where S is the sequence of entries of D in the order that they are first probed.

   (d) Bob executes the query sequence Q using $R_1^\star, \ldots, R_k^\star$ as random coin tosses. As Bob simulates all these queries, Bob records entries of D in the order they are probed to compute S. Suppose Bob encounters a new entry of D that was previously unprobed and this is the i-th probed entry. Then, Bob checks whether i appears in T or not. If $i \in T$, then Bob uses the next $b-1$ bits to decode the contents of the non-zero entry. Otherwise, if $i \notin T$, then Bob knows that the entry of D contains 0^b. As a result, Bob may execute Q to find $r/128$ unique non-zero entries of D.

   (e) Finally, Bob iterates through all parts for which the non-zero entry has not yet been discovered in the same order as Alice. For each part, Bob decodes the non-zero entry in the part using the next $\log(n/r)$ bits for its location and $b-1$ bits for its contents. So, Bob is able to successfully decode all r non-zero entries in D.

Expected length of the encoding. First, we apply Shannon's source coding theorem, which states that Alice's encoding length must be at least the entropy of $\mathbf{D}$ conditioned on Bob's input $\mathbf{R}^\star$. Note, that the distribution $\mathbf{R}^\star$ of the strings received in input by Alice and Bob is independent from $\mathbf{D}$ and thus

$$H(\mathbf{D} \mid \mathbf{R}^\star) = H(\mathbf{D}) = r\left(\log\frac{n}{r} + (b-1)\right).$$

Therefore, Alice's encoding length must be at least $r(\log(n/r)+(b-1))$ bits in expectation over $\mathbf{D}$. We next prove that the above encoding by Alice is actually smaller providing the desired contradiction.

Note that Alice's encoding length when the message starts with a 1-bit is always $1+r(\log(n/r)+(b-1)) = 1+H(\mathbf{D})$ bits. Note that the encoding starts with a 1 if $\mathtt{E}^{\mathtt{D}}(D)=1$ or if $\mathtt{E}^{\mathtt{D}}(D)=0$ and we have that $\ell > 128tk$ or $\ell_{\mathsf{nz}} < r/128$. So, the probability that Alice outputs an encoding that starts with 1 is at most

$$\Pr[\mathtt{E}^{\mathtt{D}}(D)=1] + \mathsf{Pr}[\ell > 128tk \text{ and } \mathtt{E}^{\mathtt{D}}(D)=0] + \Pr[\ell_{\mathsf{nz}} < r/128 \text{ and } \mathtt{E}^{\mathtt{D}}(D)=0].$$

We observe that $\Pr[\ell > 128tk \text{ and } \mathtt{E}^{\mathtt{D}}(D)=0] \leq \mathsf{Pr}[\ell > 128tk] \leq 1/128$, where the last inequality follows by Markov's inequality as $E[\ell] = tk$. Moreover, we have that

$$\mathsf{Pr}[\ell_{\mathsf{nz}} < r/128 \text{ and } \mathtt{E}^{\mathtt{D}}(D)=0] \leq \frac{31}{32}\cdot(1-\mathsf{Pr}[\mathtt{E}^{\mathtt{D}}(D)=1]),$$

where we have used that, by Lemma 2, $\mathsf{Pr}[\ell_{\mathsf{nz}} < r/128 \mid \mathtt{E}^{\mathtt{D}}(D)=0] \leq 31/32$. The probability that Alice outputs an encoding that starts with 1 is at most

$$\mathsf{Pr}[\mathtt{E}^{\mathtt{D}}(D)=1] + \frac{1}{128} + \frac{31}{32}\cdot(1-\mathsf{Pr}[\mathtt{E}^{\mathtt{D}}(D)=1]) = \frac{1}{128} + \frac{31}{32} + \frac{1}{32}\cdot\mathsf{Pr}[\mathtt{E}^{\mathtt{D}}(D)=1] \leq \frac{127}{128},$$

where in the last equality we used our assumption that $\mathsf{Pr}[\mathtt{E}^{\mathtt{D}}(D)=1] \leq 1/2$. Thus we can conclude that the probability α that the encoding starts with 1 is upper bound by a constant smaller than 1. That is, $\alpha \leq 127/128$.

Next, we analyze Alice's encoding when starting with a 0-bit. The hint requires r bits. As D contains at most n entries, ℓ can be encoded using $\log n$ bits. The encoding of the subset T requires $\log\binom{\ell}{r/128}$ bits. Observe that $\ell \leq 128tk = 128tr/\log n \leq 128cn$ for any constant $c > 0$. Moreover, by Stirling's approximation, we know that $\log\binom{a}{b} \leq b\log(e\cdot a/b)$ and thus T requires at most

$$\frac{r}{128}\cdot\log\left(128\cdot e\cdot\frac{\ell}{r}\right) \leq \frac{r}{128}\cdot\log\left(128^2\cdot e\cdot c\cdot\frac{n}{r}\right) = \frac{r}{128}\cdot\left(\log\frac{n}{r} - \Delta\right)$$

where $\Delta := -\log(128^2 \cdot e \cdot c)$ is a positive number that can be made arbitrarily large by picking $c > 0$ sufficiently small. Altogether, Alice's encoding starting with a 0-bit has length at most

$$1 + r + \log n + \frac{r}{128} \cdot \Big(\log \frac{n}{r} + (b-1) - \Delta\Big) + \frac{127}{128} \cdot r\Big(\log \frac{n}{r} + (b-1)\Big) = H(\mathbf{D}) + (1 + \log n + r) - \frac{r}{128}\Delta.$$

Now, since $r \geq 3 \log n$ and since by picking c sufficiently small we can make $\Delta > 0$ made arbitrarily large, we can say that, for any $\Gamma > 0$ and for sufficiently large n, we can pick c sufficiently small so that Alice's encoding starting with a 0-bit has length at most $H(\mathbf{D}) - \Gamma$.

Combining the two cases, we obtain that the expected length of Alice's encoding is, for sufficiently large n, at most

$$\alpha(1 + H(\mathbf{D})) + (1 - \alpha)(H(\mathbf{D}) - \Gamma) < H(\mathbf{D}) = H(\mathbf{D} \mid \mathbf{R})$$

by choosing $\Gamma > \alpha/(1 - \alpha)$.

Therefore, Alice's encoding contradicts Shannon's source coding theorem to complete the proof. □

4.3 Using Privacy Guarantees

By Theorem 3, for at least half of the databases D in the support of the hard distribution $\mathbf{D}$, we have $\mathtt{E}^{\mathtt{D}}(D) = 1$. We remind the reader that if $\mathtt{E}^{\mathtt{D}}(D) = 1$, then for every maximally discovering query sequence Q, the total number of probed non-zero entries will be less than $r/128$ with probability at least $31/32$.

In this section, we will construct a PPT adversary that is able to utilize the fact that $\mathtt{E}^{\mathtt{D}}(D) = 1$ for half the databases in the hard distribution $\mathbf{D}$ to break the privacy guarantees of any PIR with preprocessing scheme Π that supposedly beats our lower bound of $tr = \Omega(n \log n)$.

First, we present an algorithm **GenQueries**(D) that enables a PPT adversary to find a maximally discovering query sequence Q for any database D except with negligible probability.

GenQueries(D):

1. Set Q as an empty query sequence.
2. Repeat until the loop breaks:
   - For $q = 1, \ldots, n$:
     - Compute $H = \mathsf{Preprocess}(D, R_H)$ using random coin tosses R_H.
     - Randomly and independently execute the sequence of queries (Q, q) n times and count the number of executions cnt_q for which

       $$\big|\mathsf{contrib}(q, Q, D, H, R)\big| \geq \frac{\log n}{16}$$

       where R are random coin tosses for $|Q| + 1$ queries.
     - If $\mathsf{cnt}_q \geq n/2$, set $Q := (Q, q)$. Go back and repeat Step 2.
   - Return Q if none of the n indices satisfy the above requirement.

Lemma 3. *For any database D such that $\mathtt{E}^{\mathtt{D}}(D) = 1$, the algorithm* **GenQueries**(D) *outputs a query sequence Q that is maximally discoverable except with negligible in n probability over the internal randomness of the algorithm.*

Proof. We break the proof down into steps: one for each requirement of being maximally discoverable. We refer readers to Definition 6 for a reminder about maximally discovering query sequences.

Consider the i-th execution of the loop starting at Step 2. Suppose there exists any query $q \in \{1, \ldots, n\}$ such that

$$\Pr_{\mathbf{R}_H, \mathbf{R}}\left[\big|\mathsf{contrib}(q, Q, D, \mathbf{H}, \mathbf{R})\big| \geq \frac{\log n}{16}\right] < \frac{1}{4}.$$

By Chernoff Bounds, $\Pr[\mathsf{cnt}_q \geq n/2] \leq e^{-\Theta(n)}$ as $\mathsf{E}[\mathsf{cnt}_q] < n/4$ and each of the n runs are independent. In other words, q being added to Q occurs with negligible in n probability. As $\mathsf{E}^{\mathsf{D}}(D) = 1$, we know that any maximally discoverable query sequence has length at most $r/\log n$. By a Union Bound over at most $r/\log n$ iterations, the probability that any bad query that does not discover enough non-zero entries is added to Q is negligible in n.

Let Q be the output of **GenQueries**(D). Consider any $q \in [n]$. Suppose that

$$\Pr_{\mathbf{R}_H, \mathbf{R}}\left[|\mathsf{contrib}(q, Q, D, \mathbf{H}, \mathbf{R})| \geq \frac{\log n}{16}\right] \geq \frac{3}{4}.$$

Consider the last iteration of **GenQueries**(D) that did not append q to the query sequence Q. When considering the query q, note that $\mathsf{E}[\mathsf{cnt}_q] \geq 3n/4$. By Chernoff Bounds, we know that $\Pr[\mathsf{cnt}_q < n/2] \leq e^{-\Theta(n)}$ as each of the n runs are independent. Therefore, such a query q would have been added by **GenQueries**(D) except with negligible in n probability. By a Union Bound over all n possible queries, we know if such a $q \in [n]$ existed, **GenQueries**(D) would have added it except with negligible in n probability.

Therefore, any query sequence Q output by **GenQueries**(D) is maximally discoverable except with negligible in n probability. □

For the next step, we will use ideas from the cell sampling technique [51, 43] and the incompressibility technique [32, 31, 21, 35] that have previously been used to prove other lower bounds. In our work, we will use these ideas to construct a PPT adversary as opposed to directly proving lower bounds as done in prior works. Let us consider the following sample experiment. Recall that for any database D in the support of the hard distribution, the set $\mathsf{nz}(D) = \{j \in [n] \mid D_j \neq 0^b\}$ consists of the r indices of the non-zero entries in D. We will sample a subset T by adding each index of $\mathsf{nz}(D)$ to T with probability $1/10$. We say that a query q is *resolved* by T if all non-zero entries that were not probed by previous queries and are probed by q appear in T. Consider querying index 1 after executing any maximally discoverable query sequence Q. We know that the query to index 1 will discover at most $O(\log n)$ non-zero entries with constant probability. As a result, the query to index 1 will be resolved by a random subset T with probability $(1/10)^{O(\log n)} = 1/\mathsf{poly}(n)$. By manipulating the probabilities, we show that it is possible to find a subset T that resolves the queries to index 1 with high probability by sampling a polynomial number of random subsets T.

We formalize this idea using the following PPT algorithm that will eventually be used by a PPT adversary. We will use a slightly different method to sample subsets, but the same idea still holds.

Sample(D):

1. Let $Q \leftarrow$ **GenQueries**(D) and set $\nu = 1/(128n^{0.25})$.
2. Repeat the following $a_1 = n$ times:
   (a) Pick a uniformly random subset $T \subset \mathsf{nz}(D)$ of size exactly $r/10$.
   (b) Initialize $\mathsf{cnt} \leftarrow 0$.
   (c) Repeat the following $a_2 = n$ times:
      i. Compute $H \leftarrow \mathsf{Preprocess}(D, R_H)$ with uniformly random string R_H.
      ii. Compute $S_Q \leftarrow \mathsf{nzP}(Q, D, H, (R_1, \ldots, R_{|Q|}))$ for independently and uniformly selected sequences of coin tosses $R_1, \ldots, R_{|Q|}$.
      iii. Compute $S_1 \leftarrow \mathsf{nzP}(1, D, H, R)$ for a uniformly selected sequence of coin tosses R.
      iv. If $S_1 \setminus S_Q \subseteq T$ and $|S_{<i}| < r/128$, set $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$.
      Note that $S_i \setminus S_{<i}$ is the contribution of the query to index 1.
   (d) If $\mathsf{cnt} \geq 3/2\nu n$, return (Q, T).
3. Return $(\perp, \perp)$.

Definition 8. *Set $\nu = 1/(128n^{0.25})$ and let D be such that $\mathtt{E}^{\mathtt{D}}(D) = 1$. For a pair (Q,T) consisting of a maximally discoverable query sequence Q and a subset $T \subset \mathsf{nz}(D)$, we say $\mathtt{E}_D^{\mathtt{Q},\mathtt{T}}(Q,T) = 1$ if and only if $|T| = r/10$ and*

$$\Pr_{\mathbf{R}_H,\mathbf{R}}\left[\mathsf{contrib}(1,Q,D,\mathbf{H},\mathbf{R}) \subseteq T \wedge \left|\mathsf{nzP}(Q,D,\mathbf{H},\mathbf{R}_{\leq|Q|})\right| < \frac{r}{128}\right] \geq \nu$$

where $\mathbf{R} = (\mathbf{R}_1,\ldots,\mathbf{R}_{|Q|+1})$ is the randomness for $|Q|+1$ queries.

In what follows, we will omit the database D from $\mathtt{E}_D^{\mathtt{Q},\mathtt{T}}(Q,T)$ whenever it is clear from the context. We shall show in Lemma 5 that, except with negligible probability, **Sample**, when run on input D such that $\mathtt{E}^{\mathtt{D}}(D) = 1$, outputs a pair (Q,T) such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q,T) = 1$. We first prove the following technical lemma.

Lemma 4. *Let D be a database such that $\mathtt{E}^{\mathtt{D}}(D) = 1$ and let Q be a* maximally discovering *query sequence for D. Then,*

$$\Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}\left[\mathsf{contrib}(1,Q,D,\mathbf{H},\mathbf{R}) \subseteq \mathbf{T} \wedge \left|\mathsf{nzP}(Q,D,\mathbf{H},\mathbf{R}_{\leq|Q|})\right| < \frac{r}{128}\right] \geq 4\nu$$

where $\mathbf{T} \subset \mathsf{nz}(D)$ is a uniformly random subset of size exactly $r/10$.

Proof. Define the following variables for events:

- A denotes the event $\mathsf{contrib}(1,Q,D,H,R) \subseteq T$.
- B denotes the event $\left|\mathsf{nzP}(Q,D,H,R_{\leq|Q|})\right| < \frac{r}{128}$.
- C denotes the event $\left|\mathsf{contrib}(1,Q,D,H,R)\right| < \frac{\log n}{16}$.

With this notation in place, we need to prove that if $\mathtt{E}^{\mathtt{D}}(D) = 1$ and Q is maximally discovering for D, then it must hold that $\Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}[A \wedge B] \geq 4\nu$. We will actually prove that $\Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}[A \wedge B \wedge C] \geq 4\cdot\nu$ and then the lemma will follow from the observation that $\Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}[A \wedge B] \geq \Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}[A \wedge B \wedge C] \geq 4\cdot\nu$.

The first step consists in deriving a lower bound for $\Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}[A \mid B \wedge C]$:

$$\begin{aligned}
\Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}\Big[A \mid B \wedge C\Big] &\geq \frac{\binom{r-\log n/16}{r/10-\log n/16}}{\binom{r}{r/10}} \\
&\geq \frac{(r/10)\cdot(r/10-1)\cdots(r/10-\log n/16+1)}{r\cdot(r-1)\cdots(r-\log n/16+1)} \\
&\geq \left(\frac{r/10-\log n/16}{r}\right)^{\log n/16} \\
&\geq \left(\frac{1}{10}-\frac{\log n}{16\cdot r}\right)^{\log n/16} \qquad (\text{since } r \geq 3\log n) \\
&\geq \left(\tfrac{1}{16}\right)^{\log n/16} \geq \tfrac{1}{n^{0.25}} = 128\cdot\nu.
\end{aligned}$$

Since $\mathtt{E}^{\mathtt{D}}(D) = 1$, we have that $\Pr_{\mathbf{R}_H,\mathbf{R}}[\bar{B}] < 1/32$. By the fact that Q is maximally discoverable, we have that $\Pr_{\mathbf{R}_H,\mathbf{R}}[C] > 3/4$. Therefore, we know that $\Pr_{\mathbf{R}_H,\mathbf{R}}[B \wedge C] \geq \Pr_{\mathbf{R}_H,\mathbf{R}}[C] - \Pr_{\mathbf{R}_H,\mathbf{R}}[\bar{B}] \geq 1/32$. This implies that $\Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}[A \wedge B \wedge C] = \Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}[A \mid B \wedge C]\cdot\Pr_{\mathbf{R}_H,\mathbf{R}}[B \wedge C] \geq 4\cdot\nu$. □

We are now ready to prove the properties of algorithm **Sample**.

Lemma 5. Sample *is a PPT algorithm that, on input D such that $\mathtt{E}^{\mathtt{D}}(D) = 1$, returns a pair (Q,T) such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q,T) = 1$ except with negligible in n probability over the internal randomness of* **Sample**.

Proof. We proceed in two steps. First, we show that **Sample**(D) returns $(\perp,\perp)$ only with negligible probability. Secondly, we show that **Sample**(D) returns (Q,T) such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q,T) = 1$ except with negligible in n probability. A union bound over the two bad cases results in the desired lemma.

For the first part, note that $\mathbf{GenQueries}(D)$ returns a maximally discoverable Q except with probability negligible in n by Lemma 3. Therefore,

$$P := \Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}\Big[\mathsf{contrib}(1,Q,D,\mathbf{H},\mathbf{R}) \subseteq \mathbf{T} \wedge |\mathsf{nzP}(Q,D,\mathbf{H},\mathbf{R}_{\leq |Q|})| < \frac{r}{128}\Big] \geq 4\nu.$$

Now, for a subset T of size $r/10$, define $\mathsf{Good}(T) = 1$ iff

$$\Pr_{\mathbf{R}_H,\mathbf{R}}\Big[\mathsf{contrib}(1,Q,D,\mathbf{H},\mathbf{R}) \subseteq T \wedge |\mathsf{nzP}(Q,D,\mathbf{H},\mathbf{R}_{\leq |Q|})| < \frac{r}{128}\Big] \geq 2\nu.$$

We can bound P by conditioning on $\mathsf{Good}(T)$ in the following way

$$\begin{aligned} P \leq\ & \Pr_{\mathbf{T}}[\mathsf{Good}(\mathbf{T}) = 1] + \\ & \Pr_{\mathbf{T},\mathbf{R}_H,\mathbf{R}}\Big[\mathsf{contrib}(1,Q,D,\mathbf{H},\mathbf{R}) \subseteq \mathbf{T} \wedge |\mathsf{nzP}(Q,D,\mathbf{H},\mathbf{R}_{\leq |Q|})| < \tfrac{r}{128} \;\Big|\; \mathsf{Good}(\mathbf{T}) = 0\Big] \\ \leq\ & \Pr_{\mathbf{T}}[\mathsf{Good}(\mathbf{T}) = 1] + 2\nu. \end{aligned}$$

and thus we can conclude that $\Pr_{\mathbf{T}}[\mathsf{Good}(\mathbf{T}) = 1] \geq P - 2\nu \geq 2\nu$.

Note $\mathbf{Sample}(D)$ randomly picks $a_1 = n$ subsets T. The probability that none of the a_1 sampled T's is good is at most $(1-2\nu)^n \leq e^{\Theta(-n^{0.75})}$. Therefore, except with negligible in n probability, at least one good T is sampled. Moreover, by definition, for a good T and for randomly chosen R_H and $(R_1,\ldots,R_{|Q|+1})$, the test of the inner loop is passed with probability at least 2ν and thus $\mathsf{E}[\mathsf{cnt}] \geq 2\nu n$. By Chernoff Bounds, we know that $\Pr[\mathsf{cnt} < 3/2 \cdot \nu n]$ is negligible in n as each of the n runs are independent. This implies that $\mathbf{Sample}(D)$ returns $(\perp,\perp)$ with negligible probability.

Let us move on to the second part and consider any pair (Q,T) returned by $\mathbf{Sample}(D)$. Clearly $|T| = r/10$. Towards a contradiction, suppose that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q,T) = 0$ and consider the test of the inner loop. Note that, for such a T, $\mathsf{E}[\mathsf{cnt}] < \nu n$ and thus, by Chernoff Bounds, we get that $\Pr[\mathsf{cnt} \geq 3/2 \cdot \nu n]$ is negligible in n. Therefore, the probability that $\mathbf{Sample}(D)$ returns a pair (Q,T) where $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q,T) = 0$ is negligible. □

The previous result shows that $\mathbf{Sample}(D)$ is a PPT algorithm can find a pair (Q,T) such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q,T) = 1$ except with negligible probability. For the next theorem, we will leverage the security of PIR to show that the same statement holds for any query that is executed (not just queries to the index 1 as considered by the above statements).

To do this, we first present the following definition.

Definition 9. *Let D be a database such that $\mathtt{E}^{\mathtt{D}}(D) = 1$ and let (Q,T) be a pair such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q,T) = 1$. Then, for a sequence of random coin tosses $R = (R_H, R_1, \ldots, R_{|Q|})$ where R_H is the random coin tosses used by $\mathsf{Preprocess}$ to produce H and each R_i is the random coin tosses used by the i-th query in Q, we say that $\mathtt{E}^{\mathtt{R}}(R) = 1$, if and only if the following condition $\mathtt{Cond}(x,R) = 1$ for all $x \in \{2,\ldots,n\}$:*

- $\mathtt{Cond}(x,R) = 1$ *if and only if the following hold where $H = \mathsf{Preprocess}(D,R_H)$:*
  1. $\Pr_{\mathbf{R}_{|Q|+1}}\big[\mathsf{contrib}(x,Q,H,(R,\mathbf{R}_{|Q|+1})) \subseteq T\big] \geq \nu/2$.
  2. $|\mathsf{nzP}(Q,D,H,R)| < r/128$.

Otherwise, we denote the event $\mathtt{E}^{\mathtt{R}}(R) = 0$.

Note that $\mathtt{E}^{\mathtt{R}}(R) = 0$ if and only if there exists $x \in \{2,\ldots,n\}$ such that the $\mathtt{Cond}(x,R) = 0$ because at least one of the two conditions in the above definition is false. Note that $\mathtt{Cond}(x,R)$ is defined only for a fixed tuple (D,Q,T) such that $\mathtt{E}^{\mathtt{D}}(D) = 1$ and $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q,T) = 1$. We do not make this dependency explicit in our notation $\mathtt{Cond}(x,R)$ not to overburden our notation.

We will show next that $\mathtt{E}^{\mathtt{R}}(H,R) = 0$ cannot hold for too many choices of random sequences of coin tosses R. Otherwise, we construct a PPT adversary that will be able to compromise privacy.

Theorem 4. *Consider a (ϵ, δ)-secure PIR with preprocessing scheme with $\delta = 1/n$. Fix database D such that $\mathtt{E}^{\mathtt{D}}(D) = 1$ and let (Q, T) such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q, T) = 1$. Then*

$$\Pr_{\mathbf{R}}[\mathtt{E}^{\mathtt{R}}(\mathbf{R}) = 1] \geq \nu/4$$

where $\mathbf{R} = (\mathbf{R}_H, \mathbf{R}_1, \ldots, \mathbf{R}_{|Q|})$.

Proof. We construct a sequence of PPT adversaries $\mathcal{A}_2, \mathcal{A}_3, \ldots, \mathcal{A}_n$. If the theorem does not hold, at least one adversary in the sequence contradicts (ϵ, δ)-security of the PIR. First, we describe our PPT adversary $\mathcal{A}_j(D, H)$ for any $j \in \{2, \ldots, n\}$ as follows:

Adversary $\mathcal{A}_j(D, H)$:

1. Receive the database D and hint H.
2. Execute $(Q, T) \leftarrow \mathbf{Sample}(D)$.
3. Compute the set S_Q consisting of the non-zero entries of D that are probed by the queries of Q using hint H and random coin tosses.
4. Output challenge queries 1 and j and receive transcript $\mathcal{T}$.
5. Verify if $\mathtt{Cond}$ holds on the transcript $\mathcal{T}$ and output 1 iff it holds. Specifically,
   (a) From the transcript $\mathcal{T}$, compute the set S of the non-zero entries of D that were probed in the transcript.
   Note that $S \setminus S_Q$ is the contribution of the query for which $\mathcal{T}$ is the transcript.
   (b) If $S \setminus S_Q \subseteq T$ and $|S_Q| < r/128$, then output 1. Otherwise, output 0.

Towards a contradiction, suppose that there exists $x \in \{2, \ldots, n\}$ for which

$$\Pr_{\mathbf{R}}[\mathtt{Cond}(x, \mathbf{R}) = 1] < \nu/4$$

and consider adversary $\mathcal{A}_x$. Observe that by the property of algorithm $\mathbf{Sample}(D)$, we have that the pair computed at Step 1 satisfies $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(i, T) = 1$ except with negligible probability. For the rest of the proof, we will assume that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q, T) = 1$ as this decreases the distinguishing probability of $\mathcal{A}_x$ by at most negligible in n additive factor.

Suppose that $\mathcal{T}$ is a randomly selected transcript of the query to index 1 being executed on D. Then, by the definition of $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q, T) = 1$, we have that $\mathcal{A}_x$ outputs 1 with probability at least ν.

Consider now the case in which $\mathcal{T}$ is a randomly selected transcript of query x being executed on D and let $\mathbf{R}$ be the random of the randomness used to generate the hint and execute the queries in Q. We denote $\mathbf{H} = \mathsf{Preprocess}(D, \mathbf{R}_H)$. Then, we can analyze the probability that $\mathcal{A}_x$ outputs 1 into the scenarios when $\mathtt{Cond}(x, \mathbf{R}) = 0$ and $\mathtt{Cond}(x, \mathbf{R}) = 1$. When $\mathtt{Cond}(x, \mathbf{R}) = 1$, we can make the pessimistic assumption that $\mathcal{A}_x$ always outputs 1. For the case when $\mathtt{Cond}(x, \mathbf{R}) = 0$, we break down the analysis depending on which one of the requirements (or both) are violated. If the second requirement is violated, then $\mathcal{A}_x$ always outputs 0. If the first requirement is violated, then $\mathcal{A}_x$ outputs 1 with probability at most $\nu/2$. Therefore, when $\mathtt{Cond}(x, \mathbf{R}) = 0$, $\mathcal{A}_x$ outputs 1 with probability at most $\nu/2$. Putting it all together,

$$\begin{aligned}\Pr_{\mathbf{R}}[\mathcal{A}_x(D, \mathbf{H}) = 1] &\leq \sum_{b \in \{0,1\}} \Pr_{\mathbf{R}}[\mathcal{A}_x(D, \mathbf{H}) = 1 \mid \mathtt{Cond}(x, \mathbf{R}) = b] \cdot \Pr_{\mathbf{R}}[\mathtt{Cond}(x, \mathbf{R}) = b] \\ &\leq \tfrac{\nu}{2} + \tfrac{\nu}{4} = \tfrac{3\nu}{4}.\end{aligned}$$

Therefore the probability that $\mathcal{A}_x$ outputs 1 on a randomly chosen transcript to query x is at most $3/4\nu$. So, the difference of the probabilities that $\mathcal{A}_x$ outputs 1 in the two cases is at least $\nu/4 - \mathsf{negl}(n) = 1/(512n^{0.25}) - \mathsf{negl}(n) > 1/n$ providing our contradiction for sufficiently large n. □

4.4 Completing the Proof

In this section, we show how to utilize the property we proved in the previous section to construct an encoding of the database $\mathbf{D}$ that would beat the entropy bound, under the assumption that we have a PIR with preprocessing scheme that beats our lower bound. In Theorem 4, we proved that a PPT adversary is able to find a maximally discovering query sequence Q and subset T of size $r/10$ of non-zero entries from a database D such that any query for index $x \in [n]$ that follows the queries in Q is resolved by T with reasonably high probability. Furthermore, the query sequence Q probes at most $r/128$ non-zero entries in total. This implies that one will only need the $r/10$ entries in T and the at most $r/128$ non-zero entries probed by query sequence Q to perform all queries and thus reconstruct D. However, for this to be of use, we need to bound the error probability of the query algorithm.

Specifically, consider database D such that $\mathtt{E}^{\mathtt{D}}(D) = 1$ and (Q, T) such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q, T) = 1$. Fix a random coin toss sequence $R = (R_1, \ldots, R_{|Q|})$ with $\mathtt{E}^{\mathtt{R}}(R) = 1$. Let T be the non-zero entries probed by Q. The next lemma shows that if we query any entry $x \in [n]$ of D, the answer is correct with probability larger than $1/2$ provided that the query is resolved by subset T.

Lemma 6. *Consider a (ϵ, δ)-secure PIR with preprocessing scheme with $\epsilon \leq 1/n^2$. Let D be a database such that $\mathtt{E}^{\mathtt{D}}(D) = 1$ and let (Q, T) be such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q, T) = 1$. Then there exists a sequence of coin tosses $R = (R_H, R_1, \ldots, R_{|Q|})$ such that $\mathtt{E}^{\mathtt{R}}(R) = 1$ and, for all $x \in [n]$,*

$$\Pr_{\mathbf{R}_{|Q|+1}}\left[\mathsf{Query}(x, H, \mathbf{R}_{|Q|+1}; D) = D_x \mid \mathsf{contrib}(x, Q, H, (R, \mathbf{R}_{|Q|+1})) \subseteq T\right] > 1/2$$

where $H = \mathsf{Preprocess}(D, R_H)$.

Proof. Towards a contradiction, assume that for every R such that $\mathtt{E}^{\mathtt{R}}(R) = 1$, there exists an index $x \in [n]$ for which

$$\Pr_{\mathbf{R}_{|Q|+1}}\left[\mathsf{Query}(x, H, \mathbf{R}_{|Q|+1}; D) \neq D_x \mid \mathsf{contrib}(x, Q, H, (R, \mathbf{R}_{|Q|+1})) \subseteq T\right] \geq 1/2.$$

and thus we have

$$\begin{aligned}\Pr_{\mathbf{R}_{|Q|+1}} \quad & \left[\mathsf{Query}(x, H, \mathbf{R}_{|Q|+1}; D) \neq D_x \ \wedge \ \mathsf{contrib}(x, Q, H, (R, \mathbf{R}_{|Q|+1})) \subseteq T\right] \\ & \geq \tfrac{1}{2} \cdot \Pr_{\mathbf{R}_{|Q|+1}}\left[\mathsf{contrib}(x, Q, H, (R, \mathbf{R}_{|Q|+1})) \subseteq T\right] \geq \tfrac{\nu}{4},\end{aligned}$$

where in the last inequality we used the definition of $\mathtt{E}^{\mathtt{R}}(R) = 1$.

Consider the case where the query x is drawn uniformly at random from $[n]$. Then, for any R such that $\mathtt{E}^{\mathtt{R}}(R) = 1$, we get that

$$\Pr_{\mathbf{x},\mathbf{R}_{|Q|+1}}\left[\mathsf{Query}(\mathbf{x}, H, \mathbf{R}_{|Q|+1}; D) \neq D_x \ \wedge \ \mathsf{contrib}(\mathbf{x}, Q, H, (R, \mathbf{R}_{|Q|+1})) \subseteq T\right] \geq \frac{\nu}{4 \cdot n}.$$

Let us now look at the error probability of the PIR with preprocessing scheme when the query x is drawn uniformly at random from $[n]$. Furthermore, suppose that true randomness is used to generate the hint and execute the queries.

$$\begin{aligned}\Pr_{\mathbf{x},\mathbf{R}_H,\mathbf{R}_{\leq|Q|},\mathbf{R}} \quad & \left[\mathsf{Query}(\mathbf{x}, \mathbf{H}, \mathbf{R}; D) \neq D_{\mathbf{x}}\right] \\ & \geq \Pr_{\mathbf{R}_H,\mathbf{R}_{\leq|Q|}}[\mathtt{E}^{\mathtt{R}}(\mathbf{R}_H, \mathbf{R}_{\leq|Q|}) = 1] \Pr_{\mathbf{x},\mathbf{R}_H,\mathbf{R}_{\leq|Q|},\mathbf{R}}[\mathsf{Query}(\mathbf{x}, \mathbf{H}, \mathbf{R}; D) \neq D_{\mathbf{x}} \mid \mathtt{E}^{\mathtt{R}}(\mathbf{R}_H, \mathbf{R}_{\leq|Q|}) = 1] \\ & \geq \tfrac{\nu}{4} \cdot \tfrac{\nu}{4 \cdot n}.\end{aligned}$$

The last inequality utilizes Theorem 4 and the definition of $\mathtt{E}^{\mathtt{R}}(\mathbf{R}_H, \mathbf{R}_{\leq|Q|}) = 1$. By reminding that $\nu = 1/(128n^{0.25})$, we obtain the error probability exceeds $1/n^2$ for sufficient large n to derive a contradiction. □

Using this result about the correctness of resolved queries, we present a final impossible encoding to complete the proof of our main theorem.

Proof of Theorem 2. In our encoding scheme, Alice will receive a database D sampled according to $\mathbf{D}$. Bob will receive only Alice's encoding and will attempt to decode D correctly. Note, that Alice and Bob are computationally unbounded during the encoding and decoding. The only requirement for contradicting Shannon's source coding theorem is that the expected size of Alice's message must be strictly smaller than the entropy of random variable being encoded, $\mathbf{D}$, and that Bob must be able to decode.

Alice's Encoding. Alice receives as input database D in the support of the hard distribution $\mathbf{D}$.

1. Iterate over all possible sequences of coin tosses and all maximally discoverable query sequences to compute $\mathtt{E}^{\mathtt{D}}(D)$.
2. If $\mathtt{E}^{\mathtt{D}}(D) = 1$, then try all possible query sequences Q of length $k = /r \log n$, subsets T of size $r/10$ of $\mathsf{nzP}(D)$ and random coin tosses R to select pair (Q, T), and sequence of coin tosses $R_H, R = (R_1, \ldots, R_{|Q|})$ satisfying all the following properties:
   - $|\mathsf{nzP}(Q, D, H, R)| < r/128$.
   - For all x,
   $$\Pr_{\mathbf{R}_{|Q|+1}} \left[\mathsf{contrib}(x, Q, H, (R, \mathbf{R}_{|Q|+1})) \subseteq T\right] \geq \frac{\nu}{2}.$$
   - For all x,
   $$\Pr_{\mathbf{R}_{|Q|+1}} \left[\mathsf{Query}(x, H, \mathbf{R}_{|Q|+1}; D) = D_x \;\middle|\; \mathsf{contrib}(x, Q, H, (R, \mathbf{R}_{|Q|+1})) \subseteq T\right] > \frac{1}{2}$$

   where $H = \mathsf{Preprocess}(D, R_H)$.
3. If $\mathtt{E}^{\mathtt{D}}(D) = 0$ or a tuple of (Q, T, R_H, R) cannot be found satisfying the requirements above, then Alice will use the following naive encoding of D:
   (a) Encode a 0-bit.
   (b) Encode the location and contents of each non-zero entry using $\log(n/r) + (b-1)$ bits for a total of $r(\log(n/r) + b - 1)$ bits.
4. Otherwise, Alice will utilize the following encoding scheme leveraging the tuple (Q, T, R_H, R) as follows:
   (a) Encode a 1-bit.
   (b) Encode hint $H = \mathsf{Preprocess}(D, R_H)$ using r bits.
   (c) Encode the locations (but not the content) of the non-zero entries of D. That is, encode the r indices of $\mathsf{nz}(D)$ using $\log(n/r)$ bits for each of them.
   (d) Compute the set $S_Q := \mathsf{nzP}(Q, D, H, R)$ by executing the query sequence Q with the hint $H = \mathsf{Preprocess}(D, R_H)$ and random coin tosses R.
   (e) For each of the r non-zero entries, encode the content of the non-zero entries in the following way iterating through all r parts in some fixed order:
      - If the non-zero entry does not appear in T or S_Q, encode a 0-bit.
      - Otherwise, encode the contents of the non-zero entry using $b-1$ bits. Recall that the first bit of each non-zero entry is always a 1-bit.

Bob's Decoding. Bob receives Alice's message as input.

1. If Alice's message starts with a 0-bit, decode D trivially by reading the location and contents of each non-zero entry for each of the r parts using the next $\log(n/r) + b - 1$ bits respectively.
2. Otherwise, when Alice's message starts with a 1-bit, Bob will decode in the following way:

(a) Decode the hint H using the next r bits.

(b) Decode $\mathsf{nz}(D)$ using the next $r\log(n/r)$ bits.

(c) Decode the content of all cells in T and S_Q by iterating through all r parts in the same fixed order as Alice:

i. If the next bit is a 0, then the non-zero entry in this part is not in T or S_Q.

ii. If the next bit is a 1, then read the next $b-1$ bits to get the content of this non-zero entry.

(d) For all indices x of non-zero entries that do not appear in T or S_Q, decode the contents D_x in the following way:

i. Iterate through all possible random strings $R_{|Q|+1}$.

A. For each $R_{|Q|+1}$, try to execute x using H as the hint and $R_{|Q|+1}$ as the random coin tosses. As soon as the query attempts to probe a non-zero entry outside of T or S_Q, that is, probe a non-zero entry in the set $\mathsf{nz}(D) \setminus (T \cup S_Q)$, then Bob will skip to the next random string.

B. Note, as long as the query probes either zero entries in $\mathsf{nz}(D)$ or non-zero entries in T or S_Q, Bob can continue executing the query as the contents of all zero entries and non-zero entries in T and S_Q are known.

C. If Bob successfully finishes executing the query, Bob will keep track of all responses given by successfully executed queries.

ii. Bob computes D_x as the majority answer amongst all queries that successfully completed.

Correctness. We first show that Bob will always be able to successfully decode D correctly. In the case that Alice's message starts with a 0-bit, Bob will always be able to decode D as Alice's message naively encodes the non-zero entries. Consider the case where Alice's message starts with a 1-bit. Therefore, Alice finds the desired (Q, T, R_H, R) that satisfies the three requirements. As Alice encodes the locations of $\mathsf{nz}(D)$ and the contents and locations of T and S_Q trivially, Bob will be able to decode all of them correctly. The remaining part is that Bob must decode all non-zero entries in $\mathsf{nz}(D) \setminus (T \cup S_Q)$ correctly. For each of these remaining non-zero entries, Bob tries executing each query in $\mathsf{nz}(D) \setminus (T \cup S_Q)$ using all possible random strings with H as the hint. By the property Alice used to select the tuple (Q, T, R_H, R), Bob will be able to successfully complete each query for at least $\nu/2$-fraction of the random strings by the second requirement. Furthermore, the majority vote of the successfully completed queries will be correct by the third requirement. Therefore, Bob is able to successfully decode all non-zero entries of D.

Expected length of the encoding. Next, we will analyze the expected length of Alice's message. When Alice's message starts with a 0-bit, the length of Alice's message is $1 + r(\log(n/r) + b - 1)$ bits. For the case when Alice's message starts with a 1-bit, note that the total message length of the hint H and the set of indices $\mathsf{nz}(D)$ is at most $r + r\log(n/r)$ bits. Moreover the encoding of the contents of $\mathsf{nz}(D)$ has 1 bit for each of the r indices (this bit is 1 for indices in $T \cup S_Q$ and 0 for the other indices) and $(b-1)$ bits to specify the content of the indices in $T \cup S_Q$. As $|T| = r/10$ and $|S_Q| < r/128$, this accounts for at most $r + r \cdot (b-1)(1/10 + 1/128)$ bits. Therefore, the total number of bits used when Alice's message starts with a 1-bit is

$$1 + 2r + r\log\frac{n}{r} + r\frac{69}{640} \cdot (b-1).$$

Finally, we lower bound the probability that Alice's message starts with a 1-bit. In the case that $\mathtt{E}^{\mathtt{D}}(D) = 0$, note that Alice's message always starts with a 0-bit. By Theorem 3, we know that $\Pr_{\mathbf{D}}[\mathtt{E}^{\mathtt{D}}(\mathbf{D}) = 0] \leq 1/2$. Let us suppose that $\mathtt{E}^{\mathtt{D}}(D) = 1$. By Lemma 5, $\mathbf{Sample}(D)$ finds (Q, T) such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q, T) = 1$ except with negligible probability over the internal randomness of $\mathbf{Sample}(D)$. As a result, Alice will always be able to find (Q, T) such that $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(i, T) = 1$ by executing $\mathbf{Sample}(D)$ with all possible random strings. By Lemma 6, Alice will be able to find the necessary R_H and R such that Alice's message starts with a 1-bit

since $\mathtt{E}^{\mathtt{D}}(D) = 1$ and $\mathtt{E}^{\mathtt{Q},\mathtt{T}}(Q,T) = 1$. Therefore, Alice's message starts with a 1-bit with probability $\alpha \geq 1/2$. Putting everything together, Alice's expected message length is

$$\begin{array}{rl} & 1 + (1-\alpha) \cdot r \left[\log \frac{n}{r} + b - 1\right] + \alpha \cdot \left[2r + r \log \frac{n}{r} + r \frac{69}{640} \cdot (b-1)\right] \\ = & 1 + r \log \frac{n}{r} + r(b-1)\left[1 - \frac{571}{640}\alpha\right] + \alpha\,[2r] \\ < & r \log \frac{n}{r} + r(b-1)\left[1 - \frac{571}{640}\alpha\right] + \alpha\left[\frac{1}{3}\log n + 2r\right] \\ & \quad (\text{since, for sufficiently large } n, \text{ we have } \frac{\alpha}{3}\log n > 1) \\ < & r \log \frac{n}{r} + r(b-1)\left[1 - \frac{571}{640}\alpha\right] + \alpha\left[\frac{1}{9}r + 2r\right] \qquad (\text{since } \log n \leq r/3) \\ < & r\left[\log \frac{n}{r} + b - 1\right] = H(\mathbf{D}) \qquad (\text{since } b \geq 4 \text{ and } \alpha \geq 1/2) \end{array}$$

So, we contradict Shannon's source coding theorem and complete the proof. □

5 Conclusions

In this paper, we present a higher lower bound for computational single-server PIR with preprocessing. We show that $tr = \Omega(n \log n)$ thus improving the previously known lower bounds of $tr = \Omega(n)$ in [7]. Furthermore, our lower bounds match the highest known lower bounds in the considered model (for both cryptographic and non-cryptographic primitives).

Acknowledgements

We would like to thank Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan for pointing out an error in the previous version [58] of this work.

References

[1] I. Abraham, C. W. Fletcher, K. Nayak, B. Pinkas, and L. Ren. Asymptotically tight bounds for composing ORAM with PIR. In *IACR International Workshop on Public Key Cryptography*, pages 91–120. Springer, 2017.

[2] H. Abusalah, J. Alwen, B. Cohen, D. Khilko, K. Pietrzak, and L. Reyzin. Beyond Hellman's time-memory trade-offs with applications to proofs of space. In *ASIACRYPT'17*, pages 357–379, 2017.

[3] A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *International Colloquium on Automata, Languages, and Programming*, pages 401–407. Springer, 1997.

[4] E. Barkan, E. Biham, and A. Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In *Crypto 2006*, pages 1–21, 2006.

[5] A. Beimel and Y. Ishai. Information-theoretic private information retrieval: A unified construction. In *International Colloquium on Automata, Languages, and Programming*, pages 912–926. Springer, 2001.

[6] A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 89–98, 1999.

[7] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *Journal of cryptology*, 17(2):125–151, 2004.

[8] D. Boneh, K. Lewi, and D. J. Wu. Constraining pseudorandom functions privately. In *PKC'17*, pages 494–524, 2017.

[9] E. Boyle, J. Holmgren, and M. Weiss. Permuted puzzles and cryptographic hardness. TCC'19, 2019.

[10] E. Boyle, Y. Ishai, R. Pass, and M. Wootters. Can we access a database both locally and privately? In *Theory of Cryptography Conference*, pages 662–693. Springer, 2017.

[11] E. Boyle and M. Naor. Is there an oblivious RAM lower bound? In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 357–368, 2016.

[12] J. Brody and K. G. Larsen. Adapt or die: Polynomial lower bounds for non-adaptive dynamic data structures. *arXiv preprint arXiv:1208.2846*, 2012.

[13] 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*, pages 402–414. Springer, 1999.

[14] R. Canetti, J. Holmgren, and S. Richelson. Towards doubly efficient private information retrieval. In *Theory of Cryptography Conference*, pages 694–726. Springer, 2017.

[15] D. Cash, A. Drucker, and A. Hoover. A lower bound for one-round oblivious RAM. In *Theory of Cryptography Conference*, 2020.

[16] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 304–313, 1997.

[17] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[18] S. Coretti, Y. Dodis, and S. Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In *Crypto 2018*, pages 693–721, 2018.

[19] H. Corrigan-Gibbs and D. Kogan. The function-inversion problem: Barriers and opportunities. In *Theory of Cryptography Conference*, pages 393–421, 2019.

[20] H. Corrigan-Gibbs and D. Kogan. Private information retrieval with sublinear online time. In *Eurocrypt '20*, pages 44–75, 2020.

[21] A. De, L. Trevisan, and M. Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In *Crypto 2010*, pages 649–665, 2010.

[22] E. D. Demaine and A. López-Ortiz. A linear lower bound on index size for text retrieval. *Journal of Algorithms*, 48(1):2–15, 2003.

[23] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for private information retrieval. *Journal of Cryptology*, 14(1):37–74, 2001.

[24] 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*, pages 122–138. Springer, 2000.

[25] Y. Dodis, S. Guo, and J. Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In *Eurocrypt '17*, pages 473–495, 2017.

[26] Z. Dvir, A. Golovnev, and O. Weinstein. Static data structure lower bounds imply rigidity. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 967–978, 2019.

[27] Z. Dvir and S. Gopi. 2-server PIR with subpolynomial communication. *Journal of the ACM (JACM)*, 63(4):1–15, 2016.

[28] K. Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012.

[29] M. Fredman and M. Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 345–354. ACM, 1989.

[30] A. Gál and P. B. Miltersen. The cell probe complexity of succinct data structures. In *International Colloquium on Automata, Languages, and Programming*, pages 332–344, 2003.

[31] 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.

[32] R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 305–313. IEEE, 2000.

[33] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *International Colloquium on Automata, Languages, and Programming*, pages 803–815. Springer, 2005.

[34] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM (JACM)*, 43(3):431–473, 1996.

[35] A. Golovnev, S. Guo, T. Horel, S. Park, and V. Vaikuntanathan. Data structures meet cryptography: 3SUM with preprocessing. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 294–307, 2020.

[36] A. Hamlin, R. Ostrovsky, M. Weiss, and D. Wichs. Private anonymous data access. In *Eurocrypt'19*, pages 244–273, 2019.

[37] M. Henzinger, S. Krinninger, D. Nanongkai, and T. Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2015.

[38] P. Hubáček, M. Koucký, K. Král, and V. Slívová. Stronger lower bounds for online ORAM. In *Theory of Cryptography Conference*, pages 264–284. Springer, 2019.

[39] R. Jacob, K. G. Larsen, and J. B. Nielsen. Lower bounds for oblivious data structures. In *SODA*, pages 2439–2447. SIAM, 2019.

[40] I. Komargodski and W.-K. Lin. A logarithmic lower bound for oblivious ram (for all parameters). Cryptology ePrint Archive, Report 2020/1132, 2020. `https://eprint.iacr.org/2020/1132`.

[41] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 364–373. IEEE, 1997.

[42] 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*, pages 104–121. Springer, 2000.

[43] K. G. Larsen. The cell probe complexity of dynamic range counting. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 85–94. ACM, 2012.

[44] K. G. Larsen. Higher cell probe lower bounds for evaluating polynomials. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 293–301. IEEE, 2012.

[45] K. G. Larsen, T. Malkin, O. Weinstein, and K. Yeo. Lower bounds for oblivious near-neighbor search. In *SODA*, pages 1116–1134, 2020.

[46] 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.

[47] K. G. Larsen, M. Simkin, and K. Yeo. Lower bounds for multi-server oblivious RAMs. In *TCC (1)*, 2020.

[48] K. G. Larsen, O. Weinstein, and H. Yu. Crossing the logarithmic barrier for dynamic boolean data structure lower bounds. In *STOC 2018 Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, 2018.

[49] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[50] R. Ostrovsky and W. E. Skeith. A survey of single-database private information retrieval: Techniques and applications. In *International Workshop on Public Key Cryptography*, pages 393–411. Springer, 2007.

[51] R. Panigrahy, K. Talwar, and U. Wieder. Lower bounds on near neighbor search via metric expansion. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 805–814. IEEE, 2010.

[52] S. Patel, G. Persiano, and K. Yeo. Private stateful information retrieval. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1002–1019, 2018.

[53] S. Patel, G. Persiano, and K. Yeo. What storage access privacy is achievable with small overhead? In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 182–199, 2019.

[54] 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.

[55] M. Patrascu and E. D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM Journal on Computing*, 35(4):932–963, 2006.

[56] 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.

[57] G. Persiano and K. Yeo. Static lower bounds for non-adaptive data structures. *arXiv preprint arXiv:2001.05053*, 2020.

[58] G. Persiano and K. Yeo. Limits of preprocessing for single-server PIR. In *SODA*, 2022.

[59] S. N. Ramamoorthy and A. Rao. Lower bounds on non-adaptive data structures maintaining sets of numbers, from sunflowers. In *33rd Computational Complexity Conference*, 2018.

[60] E. Shi, W. Aqeel, B. Chandrasekaran, and B. M. Maggs. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In *CRYPTO (4)*, pages 641–669, 2021.

[61] E. Viola. Lower bounds for data structures with space close to maximum imply circuit lower bounds. *Theory of Computing*, 15(1):1–9, 2019.

[62] M. Weiss and D. Wichs. Is there an oblivious RAM lower bound for online reads? In *Theory of Cryptography Conference*, pages 603–635. Springer, 2018.

[63] A. C.-C. Yao. Should tables be sorted? *Journal of the ACM (JACM)*, 28(3):615–628, 1981.

[64] S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM (JACM)*, 55(1):1–16, 2008.

A Optimal $tr = O(n)$ Construction Sketches

In this appendix, we present constructions with $tr = O(n)$ thus matching the lower bound of [20]. These construction are implicit in [20] and we report it here for the sake of completeness.

For convenience, we present a construction with $r = O(\sqrt{n})$ and $t = O(\sqrt{n})$ that may be extended to arbitrary r-bit hints and time $t = O(n/r)$. In the offline phase, the client constructs a partition $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_m)$ of $\{1, \ldots, n\}$ into $m := \sqrt{n}$ parts each of size $\sqrt{n}$. In order to be able to succinctly represent $\mathcal{P}$, the partition is chosen to be a *pseudorandom* partition and it is selected by using a pseudorandom permutation F_K (see [49]) with a randomly selected seed K. Specifically, the j-th part $\mathcal{P}_j$ consists of $\mathcal{P}_j = \{F_K((j-1) \cdot \sqrt{n} + 1), \ldots, F_K(j \cdot \sqrt{n})\}$. Then the hint consists of bits $b_1, \ldots, b_m$ where b_i is the XOR of the entries of the database D with indices in the i-th part $\mathcal{P}_i$, and of seed K (used to reconstruct $\mathcal{P}$). Note that the length of the seed K is equal to the security parameter λ. Therefore, $r = \Theta(\sqrt{n})$.

The online phase proceeds identically to the protocol in the technical overview of [20]. Specifically, to query for entry i of D, the client finds the part $\mathcal{P}_j$ that contains i and removes from it a single index $i^\star$ determined as follows:

- With probability $1 - (\sqrt{n} - 1)/n$, the client sets $i^\star = i$;
- With the remaining probability, the client sets $i^\star$ equal to a randomly chosen index from $\mathcal{P}_j$.

The client then sends the set $P = \mathcal{P}_j \setminus \{i^\star\}$ to the server that replies with the XOR of the entries in P. Note that the server has to probe $O(\sqrt{n})$ entries to compute its replies. For the analysis of correctness and security, we refer readers back to the analysis in [20].