# Reducing the Servers' Computation in Private Information Retrieval: PIR with Preprocessing*

Amos Beimel† Yuval Ishai‡ Tal Malkin§

June 9, 2003

**Abstract**

Private information retrieval (PIR) enables a user to retrieve a data item from a database, replicated among one or more servers, while hiding the identity of the retrieved item. This problem was suggested by Chor, Goldreich, Kushilevitz, and Sudan in 1995, and since then efficient protocols with sub-linear communication were suggested. However, in all these protocols the servers' computation for *each* retrieval is at least *linear* in the size of entire database, even if the user requires only a single bit.

In this paper, we study the *computational* complexity of PIR. We show that in the standard PIR model, where the servers hold only the database, linear computation cannot be avoided. To overcome this problem we propose the model of *PIR with preprocessing*: Before the execution of the protocol each server may compute and store polynomially-many information bits regarding the database; later on, this information should enable the servers to answer each query of the user with more efficient computation.

We demonstrate that preprocessing can significantly save work. In particular, we construct for any constants $k \geq 2$ and $\epsilon > 0$: (1) a $k$-server protocol with $O(n^{1/(2k-1)})$ communication, $O(n/\log^{2k-2} n)$ work, and $O(n^{1+\epsilon})$ storage; (2) a $k$-server protocol with $O(n^{1/k+\epsilon})$ communication and work and $n^{O(1)}$ storage; (3) a *computationally-private* $k$-server protocol with $O(n^{\epsilon})$ communication, $O(n^{1/k+\epsilon})$ work, and $n^{O(1)}$ storage; and (4) a protocol with a polylogarithmic number of servers, polylogarithmic communication and work, and $O(n^{1+\epsilon})$ storage. On the lower bounds front, we prove that the product of the extra storage used by the servers (i.e., in addition to the length of the database) and the expected amount of work is at least linear in $n$. Finally, we suggest two alternative models to saving computation, by batching queries and by allowing a separate off-line interaction per future query.

**Keywords.** Privacy, Distributed Databases, Information-Theoretic Protocols, Sub-linear Computation, Sub-linear Communication.

*A preliminary version of this work appeared in [10].

†Dept. of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel. E-mail: beimel@cs.bgu.ac.il. www.cs.bgu.ac.il/∼beimel. Part of the work was done at the Division of Engineering and Applied Sciences, Harvard University, supported by grants ONR-N00014-96-1-0550 and ARO-DAAL03-92-G0115.

‡Computer Science Department, Technion, Haifa 32000, Israel. E-mail: yuvali@cs.technion.ac.il. Part of the work was done at DIMACS and AT&T Labs – Research.

§Dept. of Computer Science, Columbia University, 1214 Amsterdam Ave., New York, NY 10027, USA. E-mail: tal@cs.columbia.edu. www.cs.columbia.edu/∼tal. This work was done at AT&T Labs – Research, and at the Lab for Computer Science, Massachusetts Institute of Technology.

# 1 Introduction

As the Internet becomes ubiquitous, it is essential to protect the privacy of users. An important aspect of this problem is hiding the information a user is interested in. For example, an investor might want to know the value of a certain stock in the stock-market without revealing the identity of this stock, or a company might want to search for a certain patent without revealing the patent's identity. Towards this end, Chor, Goldreich, Kushilevitz, and Sudan [13] introduced the problem of Private Information Retrieval (PIR). A PIR protocol allows a user to access a database such that the server storing the database does not gain any information on the records the user read. To make the problem more concrete, the database is modeled as an $n$ bit string $x$, and the user has some index $i$ and is interested in privately retrieving the value of $x_i$.

Since its introduction, PIR has been an area of active research, and various settings and extensions have been considered (e.g., [1, 29, 12, 23, 19, 18, 15, 11, 8, 20, 16, 24]). Most of the initial work on PIR has focused on the goal of minimizing the *communication*, which was considered the most expensive resource. However, despite considerable success in realizing this goal, the real-life applicability of the proposed solutions remains questionable. One of the most important practical restrictions is the *computation* required by the servers in the existing protocols; in all protocols described in previous papers the (expected) work of the server(s) for each query is at least $n$, the size of the entire database. This computation overhead may be prohibitive, since the typical scenario for using PIR protocols is when the database is big.

In this paper, we initiate the study of using preprocessing to reduce server computation.[1] We demonstrate that, while without any preprocessing linear computation is unavoidable, with preprocessing and some extra storage, computation can be reduced. Such a tradeoff between storage and computation is especially motivated today; as storage becomes very cheap, the computation time emerges as the more important resource. We also provide some lower bounds on this tradeoff, relating the amount of additional storage and the computation required. Finally, we present some alternative models to saving computation. While this paper is still within the theoretical realm, we hope that the approach introduced here will lead to PIR protocols which are implemented in practice.

## 1.1 Previous Work

Before proceeding, we give a brief overview of some known results on PIR. The simplest solution to the PIR problem is that of communicating the entire database to the user. This solution is impractical when the database is large. However, if the server is not allowed to gain *any* information about the retrieved bit, then the linear communication complexity of this solution is optimal [13]. To overcome this problem, Chor et al. [13] suggested that the user accesses replicated copies of the database kept on different servers, requiring that each server gets absolutely no information on the bit the user reads (thus, these protocols are called *information-theoretic* PIR protocols). The best information-theoretic PIR protocols known prior to our work are summarized below (see Section 1.4 for some updated results): (1) a 2-server protocol with communication complexity of $O(n^{1/3})$ bits [13], (2) a $k$-server protocol, for any constant $k > 1$, with communication complexity of $O(n^{1/(2k-1)})$ bits [1] (improving on [13], see also [20, 6]), and (3) a protocol with $O(\log n)$ servers and communication complexity of $O(\log^2 n \log\log n)$ bits [4, 5, 13]. In all these protocols it is assumed that the servers do not communicate with each other. Extensions to $t$-private protocols,

[1] Gertner et al. [18] have used preprocessing in a different model, allowing to move most computation to special purpose servers (though not reducing the total work). See more below.

in which the user is protected against collusions of up to $t$ servers, have been considered in [13, 20, 6].

A different approach for reducing the communication is to limit the power of the servers; i.e., to relax the perfect privacy requirement into *computational indistinguishability* against computationally bounded servers (thus, these protocols are called *computational* PIR protocols). Kushilevitz and Ostrovsky [23], following a 2-server protocol of Chor and Gilboa [12], proved that in this setting one server suffices; under a standard number-theoretic intractability assumption they obtain, for every constant $\epsilon > 0$, a *single* server protocol with communication complexity of $O(n^\epsilon)$ bits. Essentially the same protocol can be based on any homomorphic encryption scheme [26, 30, 31]. Cachin, Micali, and Stadler [11] present a single server protocol with polylogarithmic communication complexity, based on a new number-theoretic intractability assumption called *the $\Phi$-hiding assumption.* Kushilevitz and Ostrovsky [24] construct a protocol based on a very general assumption, the existence of trapdoor permutation, with communication complexity $n\left(1 - \frac{1}{\operatorname{polylog} n}\right)$. Other works in this setting are [29, 8, 16].

Gertner, Goldwasser, and Malkin [18] were the first to address the servers' computation (see also [25]). They present a model for PIR utilizing special-purpose privacy servers, achieving stronger privacy guarantees and small computation for the original server holding the database. While their protocols save computation for the original server, the computation of the special-purpose servers (who do not hold the database) is still linear for every query. In contrast, our goal is to reduce the *total* computation by all servers. Di-Crescenzo, Ishai, and Ostrovsky [15] present another model for PIR using special-purpose servers. Their main goal is to shift most of the *communication* in PIR protocols to an off-line stage. However, by extending their technique, it is possible to shift most of the servers' *work* to an off-line stage as well, at the expense of requiring additional off-line work for each future query. This application is discussed in Section 7. Finally, Itoh [22] presents protocols slightly improving the computation of the servers compared to the protocols of [13]; however the time complexity of his protocols is still higher than $n$.

## 1.2 Our Results

As a starting point for this work, we prove that in any $k$-server protocol the total *expected* work of the servers is at least $n$. Consequently, we suggest the model of PIR with preprocessing: Before the first execution of the protocol each server computes and stores some additional information regarding the database. These bits of information are called the *extra bits* (in contrast to the original data bits). Later on, this information should enable the servers to perform less computation for each of the (possibly many) queries of the users.[2] To make this model reasonable, the number of extra bits each server is allowed to store in the preprocessing stage is polynomial in $n$.[3]

We show that preprocessing can save computation. There are three important performance measures that we would like to minimize: communication, servers' work (i.e., computation), and storage. Using different constructions, we obtain the following incomparable tradeoffs.

1. For any $\epsilon > 0$ and constant $k \geq 2$, we construct a $k$-server protocol with $O(n^{1/(2k-1)})$ communication, $O(n/(\epsilon \log n)^{2k-2})$ work, and $O(n^{1+\epsilon})$ extra bits (where $n$ is the size of the database). The importance of this protocol is that it saves work without increasing the

[2] Following the standard PIR model, we assume that each server holds a copy of the database, and define the storage complexity to include only the number of extra bits. See Remark 2.3 for further discussion.

[3] Note that if the servers were allowed to store an exponential number of extra bits then they could store the answers to every possible query in a PIR protocol without preprocessing, and their work in response to each query would simply be equal to the number of bits they have to send to the user.

communication compared to the best known information-theoretic PIR protocols.[4] We define a combinatorial problem for which a better solution will further reduce the work in this protocol.

2. Our second construction moderately increases the communication; however the servers' work is much smaller. For any constants $k \geq 2$ and $\epsilon > 0$, we construct a $k$-server protocol with polynomially many extra bits and $O(n^{1/k+\epsilon})$ communication and work.

3. We show that with a larger (polylogarithmic) number of servers, there exist protocols with polylogarithmic communication and work and with a slightly super-linear number of extra bits.

All the above protocols maintain information-theoretic user privacy, that is, the privacy is guaranteed without relying on any computational assumptions. We also present a protocol which maintains only computational privacy, that is, the privacy is guaranteed only against polynomial time servers and relies on cryptographic assumptions:

4. For any constants $k \geq 2$ and $\epsilon > 0$, assuming homomorphic encryption exists, we obtain a $k$-server protocol with polynomially many extra bits, $O(n^{\epsilon})$ communication, and $O(n^{1/k+\epsilon})$ work.

Thus, the work in the above protocol is essentially the same as the second protocol above, but the communication is much smaller.

On the negative side, we show that if the servers are only allowed to store a small number of bits in the preprocessing stage, then their computation in response to each query must be large. Specifically, we prove that the product of the expected total work done by the servers and the number of extra bits is at least linear in $n$. Our lower bounds also hold for a relaxed model, where the servers may store *arbitrary* bits of information about the database, and the storage complexity is measured as the number of stored bits in excess of $n$ (that is, in the relaxed model the servers do not necessarily hold the original database as is). However, in our lower bound we require that all servers store the same bits. The above lower bound tradeoff indicates that our upper bounds cannot be substantially improved in two extreme cases: (1) when the number of extra bits is small, and (2) when the number of servers is polylogarithmic and the work is small. When the number of *extra bits* is constant, the lower bound asserts that the work can be improved by at most a constant factor. We show that constant-factor savings can indeed be achieved, even by a 2-server protocol with a single extra bit per server. When the *work* is close to a constant (say, polylogarithmic), the bound asserts that the storage must be close to linear. Indeed, the third construction above essentially meets the lower bound in this case (however, with a polylogarithmic number of servers).

Finally, we suggest two alternative models for saving work. First, we show that sub-linear (amortized) work can be achieved by batching a large number of queries, which may possibly originate from different users. Second, we show how to shift most of the work to an off-line stage, at the cost of requiring additional off-line interaction for each future query. While generally more restrictive than our default model, both of these alternatives may be applied in the single-server case as well.

[4] As mentioned in Section 1.4, the same techniques can be used to save work also in the subsequent protocols of [9], thus still allowing to achieve the best communication complexity known to date with reduced (sublinear) work.

### 1.3 On the Difficulty of Obtaining Strong Lower Bounds

Our lower bounds on the work in PIR with preprocessing, while nearly optimal in some extreme cases, are generally very weak compared with our upper bounds. In particular, for the case where any polynomial number of extra bits is allowed to be stored, our best protocols achieve polynomial work (namely, $n^{\delta}$ work for $1/k \leq \delta \leq 1$), but we do not know of any polynomial lower bounds for this case. Below we provide some evidence to the difficulty of finding such strong lower bounds on the work, by relating it to lower bounds in the cell-probe model (which captures our understanding of the usefulness of preprocessing in general), and by pointing the relation to communication lower bounds for PIR.

**Relation to the Cell-Probe Model.** Yao's *cell-probe model* [32] provides a general framework for studying time-space tradeoffs obtainable via preprocessing (see [27] for a recent survey). The general setting considers a data-structure that enables to answer some class of queries while reading "few" bits from the data-structure to answer each query. More formally, the cell-probe problem defined by a function $f(\cdot,\cdot)$ is the following: Given an input $x$, precompute a "short" string $y$ (the data-structure) such that any query of the form $f(x,q)$ can be answered by probing "few" entries (bits by default) in $y$. The goal of the current work may be rephrased as that of finding communication-efficient PIR protocols in which the computation each server needs to perform to answer the user's query (viewed as a function $f$ of the database $x$ and the user's query $q$) admits efficient solutions in the cell-probe model.

Interestingly, the connection between cell-probe complexity and branching program complexity [28] provides evidence that proving strong lower bounds on the complexity of PIR with preprocessing (if they exist) should be difficult.

**Example 1.1** Assume that for some $\epsilon > 0$ one can rule out the existence of a 2-server protocol with $O(n^{1/3})$ communication, polynomially many extra bits, and $O(n^{2/3+\epsilon})$ work. We argue that this would imply strong lower bounds for branching programs. Fix any PIR protocol with communication $O(n^{1/3})$ and suppose that each answer bit in this protocol can be computed by a poly-size read-$r$ branching program (the variables in the branching programs are the $n$ bits of the database and the $O(n^{1/3})$ bits of the query). Then, for any fixed $x$, each server eliminates, in the pre-processing stage, all $x$-variables from the branching program, resulting in a poly-size branching program which contains only query bits. Each server can compute each answer bit by adaptively querying nodes of the branching program (each returning an $O(\log n)$-bit pointer to the next node). The work is $O(rn^{1/3}\log n)$ per answer bit (as each query variable can appear at most $r$ times on a path), and the total work is $O(rn^{2/3}\log n)$. Thus, the assumption that we can prove a lower bound as above implies that some explicit polynomial-time computable function, namely some answer bit in a $O(n^{1/3})$-communication PIR protocol, has no polynomial-size read-$r$ branching program, for $r$ as large as $n^{\epsilon}$. This should be contrasted with the fact that the largest $r$ for which a super-polynomial lower bound for read-$r$ branching program is proven is roughly $\sqrt{\log n}$ [3].

**Relation to Communication in Standard PIR.** In Section 4 we show a transformation from PIR with short communication to PIR with preprocessing, where in the preprocessing stage the server stores the answers to all possible queries, and when getting a query it simply sends the prepared answer (see Lemma 4.1). It follows, for instance, that if there is a PIR protocol with $O(\log n)$ communication then there is a PIR protocol with polynomial storage, and $O(\log n)$ communication and work. The best known lower bound for the communication in $k$-server protocols is a constant-factor improvement over the trivial $\log n$ bound, for any constant $k$ [26]. Thus, without improving lower bounds on the communication in PIR protocols with no preprocessing, one cannot hope to prove even an $\omega(\log n)$ lower bound on the work in PIR protocols with preprocessing and polynomial storage.

### 1.4 Prologue

Subsequently to this work, the communication complexity of information-theoretic PIR has been improved in [9], where it was shown that for every constant $k$ there is $k$-server protocol with communication complexity $O(n^{c \log\log k / k \log k})$ ($c$ is some constant independent of $k$). Using the same techniques as in Section 3, it is possible to reduce the work in this protocol to $n/\mathrm{polylog}\, n$ (while maintaining the same communication).

Another subsequent work which has relevance to our problem is [21]. This work provides better solutions for the batching model considered in Section 7.1, allowing to obtain single-server PIR protocols in which a batch of $t$ queries can be answered with (essentially optimal) time complexity $O(n \cdot \mathrm{polylog}\, n)$ and (essentially optimal) communication complexity $O(t \cdot \mathrm{polylog}\, n)$. However, the solutions from [21] are restricted to the case where the $t$ queries originate from the same user, whereas the amortization technique from Section 7.1 can also apply to the case where each query originates from a different user.

A different result from [21] can be used to construct PIR protocols with preprocessing in which, unlike our protocols, all three efficiency parameters are (simultaneously) slightly sublinear: communication, extra storage, and work. We note, however, that these protocols require a large (linear) number of servers.

ORGANIZATION. In Section 2 we provide the necessary definitions, in Sections 3, 4, and 5 we construct PIR protocols with reduced work, and in Section 6 we prove our lower bounds. In Section 7 we present the alternative models of batching and off-line interaction, and in Section 8 we mention some open problems.

## 2 Definitions

We start by defining the standard model for information-theoretic PIR. We restrict our attention to one-round, 1-private PIR protocols.[5]

**Definition 2.1 (PIR)** *A $k$-server PIR protocol is defined by a triple of algorithms $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$: a* query algorithm *$\mathcal{Q}(\cdot,\cdot)$, an* answer algorithm *$\mathcal{A}(\cdot,\cdot,\cdot)$, and a* reconstruction algorithm *$\mathcal{C}(\cdot,\cdot,\ldots,\cdot)$ ($\mathcal{C}$ has $k+2$ arguments). At the beginning of the protocol, the user invokes $\mathcal{Q}(n,i)$ to pick a (randomized) $k$-tuple of queries $(q_1, q_2, \ldots, q_k)$, along with an auxiliary information string* $\mathsf{aux}$.[6] *It sends to each server $\mathcal{S}_j$ the query $q_j$ and keeps* $\mathsf{aux}$ *for a later use. Each server $\mathcal{S}_j$ responds with an answer $a_j = \mathcal{A}(x, j, q_j)$. Finally, the user computes the bit $x_i$ by applying the reconstruction*

[5] All the protocols obtained in this paper, as well as all previous information-theoretic PIR protocols from the literature, use a single round of queries and answers. Our definitions may be extended to multi-round PIR in the natural way, and our lower bounds hold for the multi-round case as well. Also, in the current work we do not consider the more general case of $t$-private PIR, where the user's privacy holds against collusions of $t$ servers. Again, most of the PIR literature focuses on the special case of 1-private PIR, and most of our results extend to the general case as well.

[6] In the computational setting for PIR, defined below, the string $\mathsf{aux}$ will store trapdoor information required for efficient reconstruction. The string $\mathsf{aux}$ is also needed in some information-theoretic PIR protocols, though in all such protocols obtained in the current work it would suffice to let $\mathsf{aux} = i$.

*algorithm $\mathcal{C}(n, a_1, \ldots, a_k, \mathsf{aux})$. We assume that all algorithms are efficient in the data length $n$. We also assume, without loss of generality (see Remark 2.6), that both $\mathcal{A}$ and $\mathcal{C}$ are deterministic. A PIR protocol is secure if the following requirements hold:*

**Correctness.** *For any data length $n$, index $i$, where $1 \le i \le n$, and database $x \in \{0,1\}^n$, the user computes the correct value of $x_i$ with probability 1 (where the probability is over the randomness of $\mathcal{Q}$).*

**Privacy.** *Each server has no information about the bit that the user tries to retrieve. Formally, for every data length $n$, indices $i_1, i_2$ $(1 \le i_1, i_2 \le n)$, and server $\mathcal{S}_j$ $(1 \le j \le k)$, the distributions $\mathcal{Q}_j(n, i_1)$ and $\mathcal{Q}_j(n, i_2)$ are identical, where $\mathcal{Q}_j$ denotes the $j$-th output of $\mathcal{Q}$.*

We will also consider the relaxed notion of *computationally private* information retrieval (or *computational PIR* for short). For defining computational PIR, it is convenient to let the data length $n$ also serve as a cryptographic security parameter. A computational PIR protocol should satisfy the following relaxed privacy requirement:

**Computational privacy.** *A computationally-bounded server can only gain a negligible advantage in guessing the bit that the user tries to retrieve. Formally, for every polynomial $p(\cdot)$, data length $n$, indices $i_1, i_2$ $(1 \le i_1, i_2 \le n)$, server $\mathcal{S}_j$ $(1 \le j \le k)$, and Boolean circuit $C$ of size $p(n)$,*

$$|\Pr[C(\mathcal{Q}_j(n, i_1)) = 1] - \Pr[C(\mathcal{Q}_j(n, i_2)) = 1]| \le n^{-\omega(1)}.$$

We next define the model proposed in this paper, PIR with preprocessing. Adding the preprocessing algorithm $\mathcal{E}$ will become meaningful when we define the work in PIR protocols.

**Definition 2.2 (PIR Protocol with Preprocessing)** *A PIR protocol with $e(n)$* extra bits *$\mathcal{P} = (\mathcal{E}, \mathcal{Q}, \mathcal{A}, \mathcal{C})$ consists of four algorithms: a* preprocessing algorithm *$\mathcal{E}$ which computes a mapping from $\{0,1\}^n$ to $\{0,1\}^{e(n)}$, query and reconstruction algorithms $\mathcal{Q}$ and $\mathcal{C}$ which are the same as in standard PIR protocols, and the answer algorithm $\mathcal{A}(\cdot,\cdot,\cdot,\cdot)$, which, in addition to $x, j, q_j$, has a fourth argument – the extra bits. In an invocation of a PIR protocol with preprocessing, the string of extra bits $y = \mathcal{E}(x)$ is pre-computed and is used by the servers as the fourth input to $\mathcal{A}$. The correctness and privacy requirements are the same as in the above PIR definitions.*

**Remark 2.3** Definition 2.2 makes the implicit assumption that each server must store a full copy of the database. This may be justified by the need to allow *direct* retrieval from each server (with no privacy requirement), and by the fact that databases are commonly already replicated in the real world (e.g., for achieving fault tolerance). Thus, we consider the database replication to be given "for free," and take the storage complexity to include only the number of extra bits $|y|$. However, it is also natural to consider a relaxed model where the servers may store *arbitrary* bits of information about the database (without storing the database itself), and the storage complexity is measured as the number of stored bits in excess of $n$. Clearly, any protocol for our model can also be applied in the relaxed model. The lower bounds we will obtain in Section 6 apply also to the relaxed model.

**Definition 2.4 (Communication in PIR)** *The* communication *in a $k$-server PIR protocol is the total number of bits communicated between the user and any single server on a database of size*

*$n$, maximized over all choices of $x \in \{0,1\}^n, i \in [n]$, and all random inputs of the user. The* query complexity *is the maximal number of bits sent from the user to any single server, and the* answer complexity *is the maximal number of answer bits sent by any server.*

Next we define the work in a PIR protocol. We measure the work as in the cell-probe model: we only count the number of bits that the servers *read* (both from the database itself and from the extra bits). This definition is reasonable when dealing with lower bounds (and might even be too conservative as the work might be higher). In general this definition is not suitable for proving upper bounds. However, with the exception of the example warm-up protocol in Section 3.1, in all our protocols the servers' total time complexity (in the RAM model) is linear in the number of bits they read.

**Definition 2.5 (Work in PIR)** *Let $\mathcal{P} = (\mathcal{E}, \mathcal{Q}, \mathcal{A}, \mathcal{C})$ be a PIR protocol, and let $\mathcal{Q}_j(n, i, r)$ denote the $j$-th query $q_j$ produced by $\mathcal{Q}$ on inputs $n, i$ and a random input $r$. For a query $q$ and database $x \in \{0,1\}^n$ we denote the number of bits that $\mathcal{S}_j$ reads from $x$ and $\mathcal{E}(x)$ in response to $q$ by* $\mathrm{BITS}_j(x, q)$. *For a random input $r$, an index $i \in \{1, \ldots, n\}$, and a database $x$, the work of the servers is defined as the sum of the number of bits each server reads. Formally,* $\mathrm{WORK}(i, x, r) \stackrel{def}{=} \sum_{j=1}^{k} \mathrm{BITS}_j(x, \mathcal{Q}_j(n, i, r))$. *Finally, the work of the servers for an $i \in \{1, \ldots, n\}$, and a database $x$ is the* expected *value, over $r$, of* $\mathrm{WORK}(i, x, r)$. *That is,* $\mathrm{WORK}(i, x) \stackrel{def}{=} \mathrm{E}_r\left[\mathrm{WORK}(i, x, r)\right]$.

**Remark 2.6** In the definition of PIR we consider deterministic servers. When we consider worst case complexity, it is easy to see that this assumption is without loss of generality. This is not true when we consider average-case complexity, as in our definition of work for a PIR protocol. However, in all our protocols the servers are deterministic. Furthermore, in our lower bounds on the (average) work we do not care what the communication is, and thus we can assume that the user sends to each server a random string in addition to its query.

**Notation.** We let $[m]$ denote the set $\{1, \ldots, m\}$. For sets $A$ and $B$, we define $A \triangle B$ as the symmetric difference between the sets, namely $(A \setminus B) \cup (B \setminus A)$. For a set $A$ and an element $i$, we define $A \oplus i$ as $A \cup \{i\}$ if $i \notin A$ and as $A \setminus \{i\}$ if $i \in A$ (that is, $A \oplus i = A \triangle \{i\}$). For a finite set $A$, we define $i \in_U A$ as assigning a value to $i$ which is chosen randomly with uniform distribution from $A$ independently of any other event. We let GF(2) denote the finite field of two elements. Finally, by $H$ we denote the binary entropy function, that is, $H(p) = -p \log p - (1-p)\log(1-p)$.

## 3 Upper Bounds: A First Approach

In this section we consider variants of the information-theoretic PIR protocols of [13, 20], and obtain time-space tradeoffs for the servers' computation in these specific protocols. This approach will yield our first family of PIR protocols with preprocessing, achieving information theoretic protocols with low communication complexity and with sub-linear computation.

The rest of this section is organized as follows. In Section 3.1 we describe a simple protocol which demonstrates some of the ideas of the subsequent protocols. In Section 3.2 we present a 2-server protocol with $O(n/\log^2 n)$ work, and in Section 3.3 we generalize it to a $k$-server protocol, for a constant $k$, with $O(n/\log^{2k-2} n)$ work. In these protocols the communication is $O(n^{1/(2k-1)})$, and the storage can be made as low as $O(n^{1+\epsilon})$, for an arbitrary constant $\epsilon > 0$. In Section 3.4 we describe a combinatorial problem concerning spanning of cubes, and show that a good solution for this problem will further reduce the work in the protocols of Section 3.2 and Section 3.3.

### 3.1 A Warm-Up

We show that using $n^2/\log n$ extra bits we can reduce the work to $n/\log n$.[7] This is only a warm-up, as the communication in this protocol is $O(n)$. We save work in a simple 2-server protocol of Chor et al. [13].

ORIGINAL PROTOCOL [13]. The user selects a random set $A^1 \subseteq [n]$, and computes $A^2 = A^1 \oplus i$. The user sends $A^j$ to $\mathcal{S}_j$ for $j = 1, 2$. Server $\mathcal{S}_j$ answers with $a^j = \bigoplus_{\ell \in A^j} x_\ell$. The user then computes $a^1 \oplus a^2$ which equals $\bigoplus_{\ell \in A^1} x_\ell \ \oplus \bigoplus_{\ell \in A^1 \oplus i} x_\ell = x_i$. Thus, the user outputs the correct value. The communication in this protocol is $2(n+1) = O(n)$, since the user needs to send $n$ bits to specify (the characteristic vector of) a random subset $A^j$ to $\mathcal{S}_j$, and $\mathcal{S}_j$ replies with a single bit.

OUR CONSTRUCTION. We use the same queries and answers as in the above protocol, but use preprocessing to reduce the servers' work while computing their answers. Notice that in the above protocol each server only computes the exclusive-or of a subset of bits. To save on-line work, the servers can pre-compute the exclusive-or of some subsets of bits. More precisely, the set $[n]$ is partitioned to $n/\log n$ disjoint sets $D_1, \ldots, D_{n/\log n}$ of size $\log n$ (e.g., $D_t = \{(t-1)\log n + 1, \ldots, t \log n\}$ for $t = 1, \ldots, n/\log n$), and the exclusive-or of every subset of these sets is computed and stored by each server. That is, for every $t$, where $1 \leq t \leq n/\log n$, and every $G \subseteq D_t$, each server stores $\oplus_{\ell \in G} x_\ell$. This requires $(n/\log n) \cdot 2^{\log n} = n^2/\log n$ extra bits. Once a server has these extra bits, it can compute its answer as an exclusive-or of $n/\log n$ bits; that is, $\mathcal{S}_j$ computes the exclusive-or of the pre-computed bits $\bigoplus_{\ell \in A^j \cap D_1} x_\ell, \bigoplus_{\ell \in A^j \cap D_2} x_\ell, \ldots, \bigoplus_{\ell \in A^j \cap D_{n/\log n}} x_\ell$.

### 3.2 A 2-Server Protocol with Improved Work

We now apply an extension of the above approach to a 2-server protocol with $O(n^{1/3})$ communication.

**Theorem 3.1** *For every $\epsilon$, where $\epsilon > 4/\log n$, there exists a 2-server PIR protocol with $n^{1+\epsilon}$ extra bits in which the work of the servers is $O(n/(\epsilon^2 \log^2 n))$ and the communication is $O(n^{1/3})$.*

**Proof:** We first describe a variant of a 2-server protocol of [13], and then show how preprocessing can save work for the servers.

ORIGINAL PROTOCOL (VARIANT OF [13]). Let $n = m^3$ for some $m$, and consider the database as a 3-dimensional cube, i.e., every $i \in [n]$ is represented as $\langle i_1, i_2, i_3 \rangle$ where $i_r \in [m]$ for $r = 1, 2, 3$. This is done using the natural mapping from $[m^3]$ to $[m]^3$. In Figure 1 we describe the protocol. It can be checked that each bit, except for $x_{i_1,i_2,i_3}$, appears an even number of times in the exclusive-or the user computes in Step 3, thus cancels itself. Therefore, the user outputs $x_{i_1,i_2,i_3}$ as required. Furthermore, the communication is $O(m) = O(n^{1/3})$. Note that, while the database is considered as a 3-dimensional space, every bit of the answer is computed as the exclusive-or of a two-dimensional space.

**Our Construction.** To save on-line work the servers pre-compute the exclusive-or of some sub-cubes of bits. Let $\epsilon \geq 4/\log n$ and $\alpha = 0.5\epsilon \log n$. The set $[m]$ is partitioned to $m/\alpha$ disjoint sets $D_1, \ldots, D_{m/\alpha}$ of size $\alpha$ (e.g., $D_t = \{(t-1)\alpha + 1, \ldots, t\alpha\}$ for $t = 1, \ldots, m/\alpha$). For every $\ell \in [m]$,

[7] This example can be can be easily modified to use $O(n/\log n)$ work and $O(n^{1+\epsilon})$ storage, for any constant $\epsilon > 0$.

**A Two Server Protocol with Low Communication**

1. The user selects three random sets $A_1^1, A_2^1, A_3^1 \subseteq [m]$, and computes $A_r^2 = A_r^1 \oplus i_r$ for $r = 1, 2, 3$. The user sends $A_1^j, A_2^j, A_3^j$ to $\mathcal{S}_j$ for $j = 1, 2$.

2. Server $\mathcal{S}_j$ computes for every $\ell \in [m]$

   $$a_{1,\ell}^j \stackrel{\text{def}}{=} \bigoplus_{\ell_1 \in A_2^j, \ell_2 \in A_3^j} x_{\ell,\ell_1,\ell_2}, \qquad a_{2,\ell}^j \stackrel{\text{def}}{=} \bigoplus_{\ell_1 \in A_1^j, \ell_2 \in A_3^j} x_{\ell_1,\ell,\ell_2}, \qquad \text{and}$$
   $$a_{3,\ell}^j \stackrel{\text{def}}{=} \bigoplus_{\ell_1 \in A_1^j, \ell_2 \in A_2^j} x_{\ell_1,\ell_2,\ell},$$

   and sends the $3m$ bits $\left\{a_{r,\ell}^j \ : \ r \in \{1,2,3\}\,,\ \ell \in [m]\right\}$ to the user.

3. The user outputs $\bigoplus_{r=1,2,3}(a_{r,i_r}^1 \oplus a_{r,i_r}^2)$.

Figure 1: A two server protocol with communication $O(n^{1/3})$.

every $t_1, t_2$, where $1 \leq t_1, t_2 \leq m/\alpha$, every $G_1 \subseteq D_{t_1}$, and every $G_2 \subseteq D_{t_2}$, each server computes and stores the three bits

$$\bigoplus_{\ell_1 \in G_1, \ell_2 \in G_2} x_{\ell,\ell_1,\ell_2}, \qquad \bigoplus_{\ell_1 \in G_1, \ell_2 \in G_2} x_{\ell_1,\ell,\ell_2}, \qquad \text{and} \qquad \bigoplus_{\ell_1 \in G_1, \ell_2 \in G_2} x_{\ell_1,\ell_2,\ell}.$$

This requires $3m \cdot (m/\alpha)^2 \cdot 2^{2\alpha} \leq m^3 \cdot 2^{\epsilon \log n} = n^{1+\epsilon}$ extra bits. Once a server has these extra bits, it can compute each bit of its answer as an exclusive-or of $O(m^2/\alpha^2)$ pre-computed bits.

ANALYSIS. The answer of each server contains $O(m)$ bits, and each bit requires reading $O(m^2/\alpha^2)$ precomputed bits. Thus, the work of each server is $O(m^3/\alpha^2) = O(n/(\epsilon \log n)^2)$. Note that the identity of the precomputed bits is very easy to compute from the user's query. Thus, in this case (as well as all subsequent protocols) our measure for work (as the number of bits read) coincides with time complexity in a RAM model up to a constant factor. $\Box$

### 3.3 A $k$-Server Protocol

We now present a $k$-server generalization of the previous 2-server protocol. As in Theorem 3.1, our protocol exhibits some tradeoff between the number of extra bits and the work, without increasing communication. The work in the following protocol has a leading factor of $k^{O(k)}$ which is constant for every constant $k$. This seems to imply that this protocol can be considered practical only when $k$ is relatively small, however, as indicated in the theorem below, as long as $k \leq 0.5(\log n)^{1/4}$ the work is still $n/\text{polylog}\, n$. The exact tradeoff is described in the next theorem.

**Theorem 3.2** *For every $k$ and $\epsilon > 4k/\log n$, there is a $k$-server PIR protocol with $n^{1+\epsilon}$ extra bits in which the work is $O((2k)^{4k} n/(\epsilon \log n)^{2k-2})$ and the communication is $O(k^3 n^{1/(2k-1)})$. If $k$ is constant, the work is $O(n/(\epsilon \log n)^{2k-2})$, and if $k \leq 0.5(\log n)^{1/4}$ and $\epsilon \geq 1$ then the work is $O(n/(\epsilon \log n)^{k-2})$.*

**Proof:** We save work in a $k$-server protocol of Ishai and Kushilevitz [20].

Original Protocol [20]. As the protocol of [20] involves some notation, we only describe the protocol's relevant properties. Let $n = m^d$ for some $m$ and for $d = 2k - 1$. The database is considered as a $d$-dimensional cube. That is, every index $i \in [n]$ is represented as $\langle i_1, i_2, \ldots, i_d \rangle$ where $i_r \in [m]$ for $r = 1, 2, \ldots, d$. A sub-cube of the $d$-dimensional cube is defined by $d$ sets $A_1, \ldots, A_d$ and contains all indices $\langle i_1, i_2, \ldots, i_d \rangle$ such that $i_r \in A_r$ for every $r \in \{1, \ldots, d\}$. A sub-cube is a $(d-1)$-dimensional sub-cube if there exists some $r$ such that $|A_r| = 1$. In the protocol from [20] each server has to compute, for $k^d m$ sub-cubes of dimension $(d-1)$, the exclusive-or of bits of the sub-cube. The communication in the protocol is $O(k^3 n^{1/(2k-1)})$.

Our Construction. To save on-line work the servers compute in advance the exclusive-or of bits for some $(d-1)$-dimensional sub-cubes. Let $\alpha = \frac{\epsilon \log n}{d-1}$. Note that $\alpha \geq 2$ (since $\epsilon > 4k/\log n$ and $d = 2k-1$). The set $[m]$ is partitioned to $m/\alpha$ disjoint sets $D_1, \ldots, D_{m/\alpha}$ of size $\alpha$. For every $r \in \{1, \ldots, d\}$, every $\ell \in [m]$, every $t_1, t_2, \ldots, t_{d-1}$, where $1 \leq t_1, t_2, \ldots, t_{d-1} \leq m/\alpha$, every $G_1 \subseteq D_{t_1}$, every $G_2 \subseteq D_{t_2}$, $\ldots$, and every $G_{d-1} \subseteq D_{t_{d-1}}$, each server computes and stores the bit $\bigoplus_{\ell_1 \in G_1, \ldots, \ell_{d-1} \in G_{d-1}} x_{\ell_1, \ldots, \ell_{r-1}, \ell, \ell_r, \ldots, \ell_{d-1}}$. This requires $dm \cdot (m/\alpha)^{d-1} \cdot 2^{(d-1)\alpha} < m^d \cdot 2^{(d-1)\alpha} = n \cdot 2^{(d-1)\frac{\epsilon \log n}{d-1}} = n^{1+\epsilon}$ extra bits (the inequality holds since $d^{1/(d-1)} < 2$ and since $\alpha \geq 2$). Once a server has these extra bits, it can compute each exclusive-or of the bits of any $(d-1)$-dimensional sub-cube as an exclusive-or of $O(m^{d-1}/\alpha^{d-1})$ pre-computed bits.

Analysis. The answer of each server requires computing the exclusive-or of the bits of a $(d-1)$-dimensional sub-cube for $O(k^d m)$ sub-cubes, and each sub-cube requires reading $O((m/\alpha)^{d-1})$ bits. Thus, the number of bits that each server reads is $O(k^d m^d / \alpha^{d-1})$. Recall that $d = 2k-1$, thus the work reduces to $O((2k)^{4k} n / (\epsilon \log n)^{2k-2})$. □

### 3.4 Can the Protocols be Improved?

We now describe a combinatorial problem concerning spanning of cubes. This problem is a special case of a more general problem posed by Dodis [17]. Our protocols in Section 3.2 and Section 3.3 are based on constructions for this problem; better constructions will enable to further reduce the work in these protocols.

We start with some notation used to define the combinatorial problem. Consider the collection of all $d$-dimensional sub-cubes $\mathcal{F}_d \stackrel{\text{def}}{=} \{G_1 \times \ldots \times G_d \ : G_1, \ldots, G_d \subseteq [m]\}$ . The exclusive-or of subsets of $[m]^d$ is defined in the natural way: For sets $S_1, \ldots, S_t \subseteq [m]^d$, the point $\ell \in [m]^d$ is in $\bigoplus_{j=1}^{t} S_j$ if and only if $\ell$ is in an odd number of sets $S_j$.

**Definition 3.3 ($q$-xor basis)** *A collection $\mathcal{X} \subseteq 2^{[m]^d}$ is a $q$-xor basis of $\mathcal{F}_d$ if every sub-cube in $\mathcal{F}_d$ can be expressed as the exclusive-or of at most $q$ sets from $\mathcal{X}$.*

For example, let $D_1, \ldots, D_{m/\log m}$ be the partition of $[m]$, defined in Section 3.2, into equal size sets. The collection $\mathcal{X}_0 \stackrel{\text{def}}{=} \{G_1 \times G_2 \ : \ \exists i, j \ G_1 \subseteq D_i, G_2 \subseteq D_j\}$ is an $m^2/\log^2 m$-xor basis of $\mathcal{F}_2$. We next show how to use a $q$-xor basis of $\mathcal{F}_2$ for 2-server PIR protocols. A similar claim holds for $q$-xor basis of $\mathcal{F}_{2k-2}$ for $k$-server PIR protocols.

**Lemma 3.4** *If there is a $q$-xor basis $\mathcal{X}$ of $\mathcal{F}_2$ then there exists a 2-server PIR protocol in which the communication is $O(n^{1/3})$, the work is $O(n^{1/3} q)$, and the number of extra bits is $O(n^{1/3} |\mathcal{X}|)$.*

**Proof:** We start with the protocol of [13], described in Figure 1, in which $n = m^3$. For each set $S \in \mathcal{X}$, each server computes and stores $3m$ bits: for every $\ell \in [m]$ it stores the bits

$$\bigoplus_{(\ell_1,\ell_2)\in S} x_{\ell,\ell_1,\ell_2}, \quad \bigoplus_{(\ell_1,\ell_2)\in S} x_{\ell_1,\ell,\ell_2}, \text{ and } \bigoplus_{(\ell_1,\ell_2)\in S} x_{\ell_1,\ell_2,\ell}.$$

Recall that in the protocol of [13] each server has to compute the exclusive-or of the bits of a 2-dimensional sub-cube for $O(m)$ sub-cubes. Each exclusive-or requires reading at most $q$ stored bits, hence the total work per server is $O(mq) = O(n^{1/3}q)$.[8] □

Lemma 3.4 suggests the following problem:

**The combinatorial Problem.** *Construct a $q$-xor basis of $\mathcal{F}_d$ of size* $\mathrm{poly}(m^d)$ *such that $q$ is as small as possible.*

The next lemma shows that $q$ cannot be too small, it is at least $\Omega(m/\log m)$. In particular, the $m/\log m$-xor basis of $\mathcal{F}_1$ which we constructed in Section 3.1 is optimal up to a constant factor.

**Lemma 3.5** *If there is a $q$-xor basis of $\mathcal{F}_d$ whose size is* $\mathrm{poly}(m^d)$, *then* $q = \Omega(m/\log m)$.

**Proof:** Let $\mathcal{X}$ be a $q$-xor basis of $\mathcal{F}_d$, and assume that $|\mathcal{X}| \le m^{dc}$ for some constant $c$. Every $d$-dimensional sub-cube must be expressed as the exclusive-or of $t$ sets from $\mathcal{X}$, where $t \le q$. (However, the exclusive-or of some choices of $t$ sets is not necessarily a $d$-dimensional sub-cube.) The number of ways to choose $t$ sets is

$$\sum_{t=0}^{q} \binom{|\mathcal{X}|}{t} \le \sum_{t=0}^{q} \binom{m^{dc}}{t} \le q m^{dcq} = 2^{O(dcq \log m)}.$$

Thus, $2^{O(dcq\log m)} \ge |\mathcal{F}_d| = 2^{md}$, and since $c$ is constant, $q = \Omega\left(\frac{m}{\log m}\right)$. □

Thus, the smallest $q$ for which there is a polynomial size $q$-xor basis of $\mathcal{F}_d$ satisfies $\Omega(m/\log m) \le q \le O(m^d/\log^d m)$. We do not know where in this range the minimum $q$ lies. A construction with a smaller $q$ than the current upper bound will further reduce the work in PIR protocols without increasing the communication.

# 4 Upper Bounds: A Second Approach

In this section we present our second family of PIR protocols with preprocessing, which has the best work that we achieve, while maintaining sub-linear communication. The protocols we construct in this section are based on the following simple observation. Suppose that we have a PIR protocol with a *logarithmic* query length and a sub-linear answer length. Then it is feasible for the servers to compute and store in advance the answers to all of the (polynomially many) possible queries. When a server receives a query it only needs to read the prepared answer bits. In general,

**Lemma 4.1** *If there is a $k$-server PIR protocol in which the length of the query sent to each server is $\alpha$ and the length of the answer of each server is $\beta$, then there is a $k$-server PIR protocol with $k \cdot \beta$ work, $k(\alpha + \beta)$ communication, and $k \cdot 2^\alpha \cdot \beta$ extra-bits.*

In the rest of this section we instantiate the above lemma with different constructions of PIR protocols with logarithmic query length. Section 4.1 addresses the case of a constant number of servers (containing an information theoretic and a computational protocol), and Section 4.2 concerns the case where the number of servers is at least poly-logarithmic in $n$.

[8]Each server should be able to efficiently decide which $q$ bits it needs for computing each answer bit; otherwise our measure of work may be inappropriate.

## 4.1 Protocols with a Constant Number of Servers

The following lemma, from [6], gives upper bounds on the length of the answers in PIR protocols with a constant number of servers and a logarithmic query length.

**Lemma 4.2 ([6])** *For any integer $k \geq 2$ and constant $0 < \theta \leq 1/2$, there exists a $k$-server PIR protocol with $(k-1)(1/H(\theta) + o(1)) \log n$ query bits per server and $n^{H(\theta/k)/H(\theta)+o(1)}$ answer bits. In particular, for any constants $k \geq 2$ and $\epsilon > 0$ there exists a $k$-server protocol with $O(\log n)$ query bits and $O(n^{1/k+\epsilon})$ answer bits.*

Applying the transformation of Lemma 4.1, we obtain:

**Theorem 4.3** *For any constants $k \geq 2$ and $0 < \theta \leq 1/2$, there exists a $k$-server PIR protocol with $n^{H(\theta/k)/H(\theta)+o(1)}$ communication and work, and $n^{(k-1+H(\theta/k))/H(\theta)+o(1)}$ extra bits. In particular, for any constants $k \geq 2$ and $\epsilon > 0$ there exists a $k$-server protocol with $O(n^{1/k+\epsilon})$ communication and work and polynomially many extra bits.*

If we do not care about the exact number of extra bits and only require it to be polynomial in $n$, Theorem 4.3 gives our best upper bounds on the attainable work. However, better general tradeoffs between the storage and work can be obtained by extending the underlying family of PIR protocols. We rely on the following lemma from [13].

**Lemma 4.4 ([13])** *Suppose that there is a $k$-server PIR protocol in which the user sends $\alpha(n)$ bits to each server and receives $\beta(n)$ bits in return. Then, for any $1 \leq m(n) \leq n$, there is a PIR protocol in which the user sends $\alpha(m(n))$ bits to each server and receives $\lceil n/m(n) \rceil \beta(m(n))$ bits in return.*

We apply Lemma 4.4 to the PIR protocols of Lemma 4.2 with $m(n) = n^{\mu}$, where $0 \leq \mu \leq 1$ is a parameter. Using Lemma 4.1, we obtain the following generalized family of PIR protocols with preprocessing:

**Theorem 4.5** *For any constants $k \geq 2$, $0 < \theta \leq 1/2$, and $0 \leq \mu \leq 1$, there is a $k$-server PIR protocol with $n^{\mu H(\theta/k)/H(\theta)+(1-\mu)+o(1)}$ communication and work and $n^{\mu(k-1+H(\theta/k))/H(\theta)+(1-\mu)+o(1)}$ extra bits.*

Note that in the case $\mu = 1$, Theorem 4.5 coincides with Theorem 4.3. The other extreme case, where $\mu = 0$, roughly corresponds to the naive protocol in which the entire database is being sent to the user (requiring no storage and linear communication and work). Thus, the parameter $\mu$ may be viewed as defining a convex combination of these two extreme cases.

### 4.1.1 Choosing the Optimal Parameters for $k = 2$

Theorem 4.5 involves two parameters, $\theta$ and $\mu$. It does not specify how these parameters should be chosen, and does not explicitly specify the best obtainable tradeoff between work and storage. In the following we provide an explicit analysis of this tradeoff for the 2-server case.

Suppose that we require $n^{\tau+o(1)}$ work (and communication) for some $\tau > 0.5$. This imposes the constraint $\mu H(\theta/2)/H(\theta) + (1-\mu) = \tau$, or equivalently

$$\frac{\mu}{H(\theta)} = \frac{1-\tau}{H(\theta) - H(\theta/2)}.$$

| Comm./Work | Storage | $\theta$ | $\mu$ |
|---|---|---|---|
| 1 | 1 | – | 0 |
| 0.9 | 1.27 | $\theta_0$ | 0.32 |
| 0.8 | 1.54 | $\theta_0$ | 0.64 |
| 0.7 | 1.80 | $\theta_0$ | 0.96 |
| 0.65 | 2.04 | 0.2 | 1 |
| 0.6 | 3.20 | 0.075 | 1 |
| 0.575 | 6.45 | 0.025 | 1 |
| 0.55 | 37.7 | 0.0027 | 1 |
| 0.525 | 19235 | 0.0000026 | 1 |

Table 1: Numerical examples. The first column is the exponent of communication and work in the protocol (where, for example, 0.9 means that the communication and work is $n^{0.9+o(1)}$), the second column is the exponent of the number of bits, and the third and fourth columns are the optimal values of $\theta$ and $\mu$ for the protocol of Theorem 4.5.

Under this constraint and the additional constraint $0 \leq \mu \leq 1$ we want to minimize the storage, i.e., to minimize

$$\frac{\mu(1+H(\theta/2))}{H(\theta)} + (1-\mu) = \tau + \frac{\mu}{H(\theta)} = \tau + \frac{1-\tau}{H(\theta)-H(\theta/2)}.$$

This is equivalent to maximizing $H(\theta) - H(\theta/2)$ under the constraint $\mu = (1-\tau)H(\theta)/(H(\theta) - H(\theta/2)) \leq 1$. Let $\theta_0 \stackrel{\text{def}}{=} 1 - 1/\sqrt{2} \approx 0.293$ and $\tau_0 = H(\theta_0/2)/H(\theta_0) \approx 0.689$. The function $H(\theta) - H(\theta/2)$ increases in the interval $0 < \theta \leq \theta_0$ and decreases in the interval $\theta_0 \leq \theta \leq 0.5$. If $\tau_0 \leq \tau \leq 1$ then

$$\mu = \frac{(1-\tau)H(\theta_0)}{H(\theta_0)-H(\theta_0/2)} \leq \frac{(1-\tau_0)H(\theta_0)}{H(\theta_0)-H(\theta_0/2)} = 1.$$

Therefore, the optimal storage in this case is obtained when $\theta = \theta_0$ and is $n^{\tau+(1-\tau)/.271\ldots+o(1)} = n^{3.682\ldots-\tau\cdot 2.682\ldots+o(1)}$. It is therefore possible to get the storage arbitrarily close to linear while maintaining polynomial savings in computation. If $0.5 < \tau \leq \tau_0$ we must choose $\theta < \theta_0$ and the optimal storage is obtained when $\mu = 1$ and $\theta$ is such that $H(\theta/2)/H(\theta) = \tau$; the storage is then $n^{\tau+1/H(\theta)+o(1)}$. Table 1 contains some numerical examples of the storage for various values of $\tau$.

### 4.1.2 A Computationally-Private Protocol with Improved Communication

All of the above protocols satisfy the information-theoretic notion of PIR. We show that an asymptotically better communication complexity can be obtained in the computational model.

We rely on the PIR protocol of [6], whose properties are summarized by Lemma 4.2. A small modification of this protocol, described in [7], allows the user to reconstruct $x_i$ by probing a small number of bits from the answers.

**Lemma 4.6 ([7])** *There exists a PIR protocol with the parameters of Lemma 4.2, in which the user needs to read only $n^{\theta/H(\theta)+o(1)}$ bits from the answer of each server (the location of which is determined by the retrieval index $i$). In particular, for any integer $k \geq 2$ and constant $\epsilon > 0$, there exists a $k$-server protocol with $O(\log n)$ query bits and $O(n^{1/k+\epsilon})$ answer bits, in which the user needs to read only $O(n^\epsilon)$ bits from each answer.*

We now show how to use the above PIR protocol to achieve a low-communication PIR protocol with preprocessing. As before, the server prepares the answers to all possible queries in the preprocessing stage. When a server receives a query, it reads the prepared answer bits, but does not directly send them to the user. Instead, for each answer bit that the user needs, the user and the appropriate server execute a single-server computational protocol in which the database is the answer in the original protocol. Thus, the user and each server have to execute $O(n^{\epsilon})$ single-server (computational) protocols. Using the single-server protocol of [23] and its generalization in [26, 30, 31], whose communication complexity is $O(n^{\delta})$ for an arbitrarily small $\delta > 0$, we have:

**Theorem 4.7** *Suppose that homomorphic encryption exists.*[9] *Then, for any constants $k \geq 2$ and $\epsilon > 0$ there exists a $k$-server computational PIR protocol with polynomially many extra bits, $O(n^{\epsilon})$ communication, and $O(n^{1/k+\epsilon})$ work.*

### 4.2 Protocols with a Polylogarithmic Number of Servers

We now consider the case where the number of available servers is at least logarithmic in $n$, and show that in this case there exist protocols with polylogarithmic work and polynomial storage. Note that the previous construction from Section 4.1 is insufficient for this purpose. Indeed, the length of each query in the PIR protocol of Theorem 4.2 is at least $(k-1)\log n$ and the length of the answers is at least $n^{1/k}$; it is thus impossible in that protocol for the answer length to be polylogarithmic while maintaining logarithmic query length (which translates into polynomial storage).

PIR protocols with polylogarithmic number of servers, logarithmic query length, and polylogarithmic answer length may be obtained from the following lemma (which optimizes previous constructions from [13, 5, 2]).

**Lemma 4.8 ([15, 6])** *Let $m$ and $d$ be positive integers such that $\binom{m+d}{d} \geq n$. Then, there exists a $(d+1)$-server PIR protocol with $\alpha = \lceil \log(d+2) \rceil m$ query bits and a single answer bit per server.*

Substituting $m = \frac{\log n}{\epsilon \log\log n}$ and $d = m \log^{\epsilon} n = \frac{\log^{1+\epsilon} n}{\epsilon \log\log n}$, and applying Lemma 4.1, we get:

**Theorem 4.9** *For any constant $\gamma > 0$ there is an $O(\frac{\log^{1+\gamma} n}{\log\log n})$-server PIR protocol with $O(\frac{\log^{2+\gamma} n}{\log\log n})$ communication and work, and $\tilde{O}(n^{1+1/\gamma})$ extra bits.*

In Theorem 6.5 we will show that the product of the work and the number of extra bits must be at least linear in $n$. It follows that the number of extra bits in the protocol of Theorem 4.9 is not far from optimal when $\gamma$ is large.

## 5 Reducing the Work with One Extra Bit

In this section we show that even a single extra bit is helpful. Specifically, we describe a $k$-server protocol with one extra bit and $\frac{k}{2k-1} \cdot n$ work. The work in this protocol is close to optimal, since in Theorem 6.5 we will show that, for any $k$, the work in any $k$-server PIR protocol with one extra bit must be at least $n/4$. Our protocol is thus nearly optimal as the work approaches $n/2$ as the

[9] Roughly speaking, a homomorphic encryption scheme is a probabilistic encryption together with an efficient algorithm Hom such that: (1) the message domain is some Abelian group, and (2) given two encryptions $c_1 = \mathrm{Enc}_k(m_1, r_1)$ and $c_2 = \mathrm{Enc}_k(m_2, r_2)$, the algorithm Hom outputs $c = \mathrm{Hom}(c_1, c_2)$ such that $c$ is a valid encryption of $m_1 + m_2$. Intuitively, Hom "sums" the encrypted messages. For example, homomorphic schemes exist under the decisional Diffie-Hellman assumption and under the quadratic-residuosity assumption. See, e.g., [30] for more details.

**A Protocol with One Extra Bit**

**Servers' input:** a database $x \in \{0,1\}^n$ and an extra-bit $b = \oplus_{i=1}^n x_i$.

**User's input:** an index $i \in [n]$.

**User:**

- Randomly partition the set $[n] \setminus \{i\}$ into $k$ disjoint sets $A_1, \ldots, A_k$.
- Choose a random number $h \in_U \{1, \ldots, 2k-1\}$.
- If $h \leq k$ then send $A_h \cup \{i\}$ to $\mathcal{S}_h$ and send $\perp$ to $\mathcal{S}_1, \ldots, \mathcal{S}_{h-1}, \mathcal{S}_{h+1}, \ldots, \mathcal{S}_k$ (where $\perp$ is a special symbol).
- Otherwise (i.e., $h > k$), send $A_j$ to $\mathcal{S}_j$ for every $j = 1, \ldots, k$.

**Server $\mathcal{S}_j$ upon reception of a set $B_j \neq \perp$:** send the bits $\{x_\ell : \ell \in B_j\}$ to the user. In addition, Server $\mathcal{S}_1$ sends the extra bit $b$ to user.

**User:** If $h \leq k$ then the user has received the bit $x_i$. Otherwise, it has received $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ and $b = \oplus_{\ell=1}^n x_\ell$, and can reconstruct $x_i$.

Figure 2: A $k$-server protocol with one extra bit and $k/(2k-1) \cdot n$ work.

number of servers grows. However, the communication in this protocol is linear in the size of the database. The purpose of presenting this protocol is to show that even with one extra bit things become non-trivial, thus giving some explanation why the proof of the lower bound in Theorem 6.5 is somewhat complicated, and to show that our lower bounds for constant number of extra bits are close to optimal in terms of the tradeoff between work and storage.

**Theorem 5.1** *There is a $k$-server PIR protocol with* one *extra bit, $\frac{k}{2k-1} \cdot (n+1)$ work, and $O(kn)$ communication.*

**Proof:** We prove the theorem by describing in Figure 2 a protocol satisfying its conditions. In this protocol the extra bit is $b \stackrel{\text{def}}{=} \oplus_{\ell=1}^n x_\ell$ and it is held by $\mathcal{S}_1$.

We next analyze the properties of this protocol.

**Privacy.** Fix $j$. Server $\mathcal{S}_j$ gets no information if $B_j = \perp$. We prove that $\Pr[\ell \in B_j | B_j \neq \perp] = 1/k$ for every $\ell$, implying that $\mathcal{S}_j$ gets no information about $i$ when $B_j \neq \perp$. For $\ell \neq i$, index $\ell$ is equally likely to be in any $A_1, \ldots, A_k$, thus the probability that it belongs to $B_j$, provided that $B_j \neq \perp$, is $1/k$. For $\ell = i$:

$$\Pr[i \in B_j | B_j \neq \perp] = \frac{\Pr[i \in B_j \wedge B_j \neq \perp]}{\Pr[B_j \neq \perp]} = \frac{1/(2k-1)}{1/(2k-1) + (k-1)/(2k-1)} = \frac{1}{k}.$$

Hence, $\mathcal{S}_j$ gets no information about $i$. This holds for every $j$, and the privacy follows.

**Work.** With probability $(k-1)/(2k-1)$ the work is $n$ and with probability $k/(2k-1)$ the expected work is $1 + (n-1)/k$. Thus, the expected work is:

$$\frac{k-1}{2k-1} \cdot n + \frac{k}{2k-1}\left(1 + \frac{n-1}{k}\right) < \frac{k(n+1)}{2k-1}.$$

Thus, for $k = 2$ the work is $2n/3$ and it asymptotically approaches $n/2$ as $k$ grows.

**Communication.** We naively implement the protocol such that the user sends the set $A_j$ using $A_j$'s characteristic vector of length $n$. The answer of $\mathcal{S}_j$ is a string of length $|A_j|$. Thus, the communication is $O(kn)$. □

Finally, we show that we can still save some work with one extra bit even if we want the communication to be sub-linear; however, we only save an *additive* term of the order of the communication complexity.

**Corollary 5.2** *For every $\delta$, where $1/2 \leq \delta \leq 1$, there exists a 2-server PIR protocol with one extra bit, $n - \Theta(n^\delta)$ work, and $O(n^\delta)$ communication.*

**Proof:** Partition the $n$ bits of the database into two parts, one part $1, \ldots, n^\delta$ of size $n^\delta$, and the other part $n^\delta + 1, \ldots, n$ of size $\hat{n} = n - n^\delta$. The bit that the user wants is in one of the parts; the user chooses an arbitrary bit from the other part. The user and the server execute the PIR protocol of Theorem 5.1 on the first part, and the 2-server PIR protocol of [13] with communication $O(\hat{n}^{1/2})$ and expected work $\hat{n}$ on the second part,[10] each execution with the appropriate bit (the bit that the user wants and the arbitrary bit it chose). The expected work is $\frac{2}{3} \cdot n^\delta + n - n^\delta = n - \frac{1}{3} \cdot n^\delta$. The communication is $O(n^\delta) + (n - n^\delta)^{1/2} = O(n^\delta)$. Finally, only a single extra bit is required for preprocessing the first part of the database. □

# 6 Lower Bounds

In this section we obtain lower bounds on the complexity of PIR with preprocessing. We first prove that without extra bits the expected number of bits all servers must read together is at least $n$, the size of the database. We then prove that if there are $e$ extra bits ($e \geq 1$) then the expected number of bits all servers must read is $\Omega(n/e)$. These lower bounds hold for any number of servers $k$, and regardless of the communication complexity of the protocol. Furthermore, the lower bounds also hold for the relaxed model, discussed in Remark 2.3, where the servers may store *arbitrary* $n + e$ bits of information about the database, and the storage complexity is measured as the number of stored bits in excess of $n$. However, in our lower bound we require that all servers store the same string of $n + e$ bits.

Note that we only prove that the *expectation* is big, since there could be specific executions where the servers read together less bits.[11] The fact that there are executions with small work should be contrasted with single-server (computational) PIR protocols without preprocessing, where each server stores the database as is. In this case the server has to read the entire database for each query, except with negligible probability: if the server does not read $x_i$ in response to some query, it knows that the user is not interested in $x_i$, violating the user's privacy.

To prove the lower bounds, we fix a PIR protocol $\mathcal{P}$, and introduce some notation referring to this protocol. Let $m = n + e$, that is, $m$ is the number of bits held by each server. We assume that each server holds the same encoding of the database, and $\mathcal{E}$ denotes the encoding function, mapping from $\{0,1\}^n$ to $\{0,1\}^m$. The notation $P^j(h, x)$ denotes the probability that server $\mathcal{S}_j$ on database $x$ reads the $h$-th bit of $\mathcal{E}(x)$ when the retrieval index is $i = 1$. This probability is taken over the

[10] They cannot use the better protocol of [13] with communication $O(\hat{n}^{1/3})$ since the work in this protocol is $3\hat{n}/2$.

[11] For example, consider the following 2-server protocol (without any extra bits). The user with probability $\frac{1}{n}$ sends $i$ to server $\mathcal{S}_1$, and nothing to server $\mathcal{S}_2$, and with probability $(1 - \frac{1}{n})$ sends a random $j \neq i$ to $\mathcal{S}_1$, and sends $[n]$ to $\mathcal{S}_2$. Server $\mathcal{S}_j$, upon reception of a set $B_j$, replies with the bits $\{x_\ell : \ell \in B_j\}$ to the user.

user's random input $r$. Next, define $P(h,x) \stackrel{\text{def}}{=} \sum_{j=1}^{k} P^j(h,x)$, and $P(h) \stackrel{\text{def}}{=} \mathrm{E}_{x \in_U \{0,1\}^n} P(h,x)$. By linearity of expectation, $P(h)$ is the sum, over all servers $\mathcal{S}_j$, of the probability that $\mathcal{S}_j$ reads the $h$-th bit of $\mathcal{E}(x)$, where this time each probability is taken over both the user's random input and the uniform distribution over the database $x$.

By the privacy requirement the probability that server $\mathcal{S}_j$ reads the $h$-th bit of $\mathcal{E}(x)$ on database $x$ is independent of the retrieval index $i$. Thus, although we defined $P^j(h,x)$ with respect to retrieval index 1, we could have chosen any index. Therefore, the same holds for $P(h,x)$ which is the sum of the probabilities $P^j(h,x)$.

**Observation 6.1** *The sum of probabilities $P(h,x)$ is independent of the retrieval index $i$.*

We next prove that $\sum_{h=1}^{m} P(h,x)$ is the expected work of the servers on database $x$.

**Claim 6.2** *For every retrieval index $i$ it holds that* $\mathrm{WORK}(i,x) = \sum_{h=1}^{m} P(h,x)$.

**Proof:** Fix any retrieval index $i$, and define the random variables $Y^j(x,1), \ldots, Y^j(x,m)$ where $Y^j(h,x) = 1$ if server $\mathcal{S}_j$ reads the $h$-th bit with database $x$ and retrieval index $i$ and $Y^j(h,x) = 0$ otherwise (these random variables depend on the query, which in turn depends on $r$ – the random input of the user). Clearly, $\mathrm{E}_r[Y^j(h,x)] = \Pr[Y^j(h,x) = 1] = P^j(h,x)$. Next define $Y(x) \stackrel{\text{def}}{=} \sum_{h \in [m]} \sum_{j=1}^{k} Y^j(h,x)$. That is, $Y$ is a random variable denoting the number of bits the servers read and $\mathrm{WORK}(i,x) = \mathrm{E}_r[Y(x)]$ (by Observation 6.1 this is true for every index $i$). The claim follows by noting that $\mathrm{E}_r[Y(x)] = \mathrm{E}_r\left[\sum_{j=1}^{k} \sum_{h \in [m]} Y^j(h,x)\right] = \sum_{h \in [m]} \sum_{j=1}^{k} \mathrm{E}_r[Y_\ell] = \sum_{h \in [m]} P(h,x)$. $\square$

To get a lower bound of $w$ on the expected work, it suffices to show that for some database $x$, $\sum_{h \in [m]} P(h,x) \geq w$. Towards this goal, we make the following definition: Let $H \subset [m]$. We say that $x$ is *$H$-dependent* if there is some $x' \neq x$ such that $\mathcal{E}(x)$ and $\mathcal{E}(x')$ agree on all coordinates in $[m] \setminus H$.

**Claim 6.3** *If $x$ is $H$-dependent then $\sum_{h \in H} P(h,x) \geq 1$.*

**Proof:** Let $x' \neq x$ be such that $\mathcal{E}(x)$ and $\mathcal{E}(x')$ agree on all coordinates in $[m] \setminus H$, and fix an index $i$ such that $x_i \neq x'_i$. If in any execution of the PIR protocol with retrieval index $i$ and database $x$ the servers read only bits from $[m] \setminus H$ then the user gets the same answers for databases $x$ and $x'$, and it errs for at least one of these databases, contradicting the correctness requirement. Thus, the probability that at least one of servers reads a bit from $H$ when the database is $x$ and the retrieval bit is $i$ is one. Since $P(h,x)$ is an upper bound on the probability that at least one of servers reads the $h$-th bit, and since $P(h,x)$ is independent of the retrieval index $i$, we conclude that $\sum_{h \in H} P(h,x) \geq 1$. $\square$

## 6.1 Lower Bound without Extra Bits

We next prove that without extra bits the expected number of bits that the servers read is at least $n$. This lower bound holds for every database.

**Theorem 6.4** *For every PIR protocol without extra bits,* $\mathrm{WORK}(i,x) \geq n$ *for every $x \in \{0,1\}^n$ and every $i \in [n]$.*

**Proof:** In this case the encoding function $\mathcal{E} : \{0,1\}^n \to \{0,1\}^n$ must be one-to-one and onto. Thus, for every $x \in \{0,1\}^n$ and every $h \in [n]$ there must be some $x' \neq x$ such that $\mathcal{E}(x)$ and $\mathcal{E}(x')$ agree on all coordinates in $[n] \setminus \{h\}$. Claim 6.3 implies that $P(h,x) \geq 1$ for every $h$, and, by Claim 6.2, $\mathrm{WORK}(i,x) \geq n$. $\square$

### 6.2 Lower Bound with Extra Bits

In this section we show that a small number of extra bits cannot reduce the work too much.

**Theorem 6.5** *For every PIR protocol with $e$ extra bits* $\mathrm{E}_x\mathrm{WORK}(i,x) \geq n/(2(e+1))$ *for every* $i \in [n]$. *In particular, there exists a database* $x \in \{0,1\}^n$ *such that* $\mathrm{WORK}(i,x) \geq n/(2(e+1))$.

**Proof:** Assume, without loss of generality, that $P(1) \geq P(2) \geq \ldots \geq P(m)$ and consider the set $H = \{n, n+1, \ldots, n+e\}$ of size $e+1$. The probability that a random database $x$ is not $H$-dependent is at most $1/2$. (The worst case is when the projection of $\mathcal{E}(\{0,1\}^n)$ on $\{1,\ldots,n-1\}$ contains $2^{n-1}-1$ words with multiplicity one, and one word with multiplicity $2^{n-1}+1$.) By Claim 6.3, for at least half of the databases it holds that $\sum_{h=n}^{n+e} P(h,x) \geq 1$. Therefore, $\sum_{h=n}^{n+e} P(h) = \sum_{h=n}^{n+e} \mathrm{E}_x P(h,x) \geq 1/2$. By the assumption that the probabilities are non-decreasing, we get that $P(h) \geq 1/(2(e+1))$ for every $1 \leq h \leq n$. Thus, by Claim 6.2 and linearity of the expectation, $\mathrm{E}_x\mathrm{WORK}(i,x) \geq \sum_{h=1}^{n+e} P(h) \geq n/(2(e+1))$ as required. □

**Remark 6.6** Theorems 6.4 and 6.5 hold (up to a negligible difference) even for computational PIR. The reason that the lower bound remains valid is that we only use the privacy requirement in the assumption that $P^j(h,x)$ is independent of the retrieval index. This assumption must hold also with respect to computational privacy (up to a negligible factor), or otherwise we get an efficient distinguisher between two indices. While we explicitly address only the (perfect) information-theoretic setting, all the lower bounds in this section hold, up to a negligible difference, for $k$-server computational PIR protocols as well. The same holds if the perfect correctness requirement is relaxed to allow negligible error probability.

## 7 Alternative Models for Saving Work

The PIR with preprocessing model allows to reduce the on-line work in PIR protocols. In this section we discuss two alternative models for achieving the same goal. While both are in a sense more restrictive than our original model, in some situations they may be preferred. For instance, they both allow to substantially reduce the on-line work in *single*-server computational PIR, an important advantage over the solutions of Sections 3–5.

### 7.1 Batching Queries

In the first alternative setting, we allow servers to batch together several queries before replying to all of them. By default, no preprocessing is allowed. The main performance measures of PIR protocols in this setting are: (1) the *amortized communication complexity*, defined as the average communication per query; (2) the *amortized work* per query; (3) the *batch size*, i.e., the minimum number of queries which should be processed together; and (4) the extra *space* required for storing and manipulating the batched queries. Note that in the case of a single user, the trivial PIR solution of communicating the entire database gives an optimal tradeoff between the batch size and the amortized work, namely their product is $n$. (This is optimal by the lower bound of Theorem 6.4.) However, this solution provides a poor tradeoff between the amortized communication and the batch size (their product is $n$). Moreover, as in the remainder of this paper, we are primarily interested in the general situation where different queries may originate from different users.

Our main tool for decreasing the amortized work is a reduction to matrix multiplication. The savings achieved by the state-of-the-art matrix multiplication algorithms can be translated into savings in the amortized work of the PIR protocols. To illustrate the technique, consider the 2-server PIR protocol described in Figure 1. In a single invocation of this protocol, each server has to compute the exclusive-or of $O(n^{1/3})$ two-dimensional sub-cubes. Each such computation can be expressed as evaluating a product (over GF(2)) of the form $a^t X_\ell b$, where $a$ and $b$ are vectors in $\mathrm{GF}(2)^{n^{1/3}}$ determined by the user's query, and $X_\ell$ is an $n^{1/3} \times n^{1/3}$ matrix determined by the database $x$. It follows that the answers to $n^{1/3}$ queries can be computed by evaluating $O(n^{1/3})$ matrix products of the form $A \cdot X_\ell \cdot B$, where the $j$-th row of $A$ and the $j$-th column of $B$ are determined by the $j$-th query. The communication complexity of the protocol is $O(n^{1/3})$ per query, and its space and time requirements depend on the matrix multiplication algorithm being employed. Letting $\omega$ denote the exponent of matrix multiplication (Coppersmith and Winograd [14] prove that $\omega < 2.376$), the amortized work can be as low as $O(n^{1/3}n^{\omega/3})/n^{1/3} = O(n^{\omega/3})$, with batch size $n^{1/3}$.

**Theorem 7.1** *There exists a 2-server amortized PIR protocol with batch size $n^{1/3}$, amortized communication $O(n^{1/3})$, and amortized work $O(n^{\omega/3})$, where $\omega$ denote the exponent of matrix multiplication.*

The same approach can also be employed towards reducing the amortized work in computational single-server PIR protocols. We use the protocols from [23, 26, 30, 31], which utilize homomorphic encryption.

In the protocols of [23, 26, 30, 31], the server's computation on multiple queries can be reduced to evaluating several matrix products. In each product one matrix depends on the queries and is given in an encrypted form (using a key held by the user) and the other depends on the database and is given in a plain form. Now, by the definition of homomorphic encryption, an encryption of the sum of two encrypted values and an encryption of the product of an encrypted value with a non-encrypted value are both easy to compute. It turns out that these two operations are sufficient for implementing a fast matrix multiplication algorithm where one of the matrices is given in an encrypted form and the output may be encrypted as well. (The protocols of [23, 26, 30] are recursive and the above statement is true only for the first level of recursion; however the work in the first level dominates the work in the entire protocol.) Thus, the next theorem follows by modifying the protocol from [23, 26, 30].

**Theorem 7.2** *If homomorphic encryption exits, then for any constant $\epsilon > 0$ there exists a constant $\delta > 0$ and a computational 1-server amortized PIR protocol with batch size $n^\epsilon$, communication $O(n^\epsilon)$, and amortized work $O(n^{1-\delta})$.*

For general homomorphic encryption schemes, the above requires that all users encrypt their query vectors using the same key (otherwise, there is no meaningful way of "adding" encrypted entries from distinct vectors). Luckily, this problem can be avoided if the homomorphic encryption scheme being employed is El-Gamal encryption, where the same multiplicative group is used by all users (yet, different users can have independent public and private keys). In this case, a fast matrix multiplication algorithm can be applied, where addition of two encrypted entries is replaced by multiplication in their common group.

### 7.2 Off-Line Interaction

In the PIR with preprocessing model, a single off-line computational effort by each server can reduce the on-line work in each of an unlimited number of future queries. It is natural to ask whether the on-line work can be further reduced if a separate off-line procedure is applied for each query. More precisely, we allow the user and the servers to engage in an off-line protocol, involving both communication and computation, so as to minimize the total on-line work associated with answering a *single* future query. (The off-line protocol may be repeated an arbitrary number of times, allowing to efficiently process many on-line queries.) During the off-line stage, the database $x$ is known to the servers but the retrieval index $i$ is unknown to the user. The goal is to obtain protocols with a small on-line work (for both the user and the servers) and "reasonable" off-line work.[12]

Towards achieving the above goal we extend an idea from [15]. Recall that in contrast to the current goal, the goal of Di-Crescenzo et al. [15] is to shift most of the *communication* and the *user's* computation to an off-line stage. A simple solution proposed in [15] is the following. Let $\mathcal{P}$ be a $k$-server (information-theoretic or computational) PIR protocol, in which the user sends at most $\alpha$ bits to each server and receives at most $\beta$ bits in return. The execution of $\mathcal{P}$ is broken into an off-line stage and an on-line stage as follows. During the off-line stage, the user picks a random retrieval index $r \in [n]$, computes a corresponding $k$-tuple of queries $(q_1, \ldots, q_k)$, and sends each query $q_j$ to the server $\mathcal{S}_j$. During the on-line stage the user, on input $i$, computes the offset $\Delta \stackrel{\text{def}}{=} i - r \pmod n$ and sends it to each server; each server $\mathcal{S}_j$ replies with the answer (according to $\mathcal{P}$) to the off-line query $q_j$ on a virtual database obtained from cyclically shifting $x$ by $\Delta$ places to the left. Finally, the user reconstructs $x_i$ by applying the reconstruction function of $\mathcal{P}$ to obtain the $r$th bit of the shifted database, which is the desired bit $x_i$.

The off-line communication in the above protocol is $O(\alpha)$, and the on-line communication is $O(\beta+\log n)$. Fortunately, most known PIR protocols admit variants in which the answer complexity $\beta$ is very small, as small as a single bit in the multi-server case, while $\alpha$ is still sub-linear (see [15] for a detailed account). For instance, in the 2-server computational PIR protocol of Chor and Gilboa [12], $\alpha = 2^{O\left(\sqrt{\log n}\right)}$ and $\beta = 1$. Substituting such a protocol for $\mathcal{P}$ in the above procedure makes the on-line communication and the total *user's* computation very small (in fact, comparable to those of non-private retrieval) while maintaining a reasonably low off-line communication.

Note, however, that the on-line work performed by the servers is still linear. The only observation that is left to make is that since there are only $n$ possible on-line queries made by the user, the servers can pre-compute the answers to each of these queries. Thus, paying at most $O(\beta n)$ storage and $O(n^2)$ off-line computation (assuming that the time complexity of $\mathcal{P}$ is $O(n)$), the on-line work is reduced to $O(\beta)$.

Finally, we note that when instantiating the above general scheme with some concrete PIR protocols (including the abovementioned 2-server protocol from [12]) the time complexity of the off-line stage can be significantly reduced. Specifically, in these cases each server needs to compute the modulo-2 inner product of a length-$n$ vector determined from the user's off-line query with *all* cyclic shifts of the database. This may be viewed as multiplication of an $n \times n$ *Toeplitz* matrix determined by the database by a vector determined from the user's query. Using the FFT algorithm, this computation can be done in nearly-linear time.

# 8 Open Problems

We have shown that using preprocessing in PIR protocols one can obtain substantial savings in the amount of computation without severely affecting the communication complexity. However, this work only initiates the study on PIR with preprocessing, and there are many open problems for

[12]This may be compared to the model of Gertner et al. [18], where they shift most computation to a "more reasonable place" (special purpose servers); here we shift most computation to a "more reasonable time" (the off-line stage).

further research. The obvious open problem is whether our upper bounds for the case of a constant number of servers can be improved:

*How much can the work be reduced using polynomially many extra bits?*
*How much can be saved using* linearly *many extra bits?*

All the solutions provided in this work (with the exception of Section 7) are multi-server PIR protocols. It is therefore natural to ask:

*Can preprocessing substantially save work in single-server PIR protocols?*

**Acknowledgments.** We thank Oded Goldreich for suggesting the question of preprocessing in PIR protocols, and Yoav Stahl for valuable comments on earlier drafts of this paper. The first author would like to thank Les Valiant for the support and hospitality while he was at Harvard. Finally, we thank the anonymous referees for their helpful comments.

## References

[1] A. Ambainis. Upper bound on the communication complexity of private information retrieval. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proc. of the 24th International Colloquium on Automata, Languages and Programming*, volume 1256 of *Lecture Notes in Computer Science*, pages 401–407. Springer, 1997.

[2] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. of the 23th Annu. ACM Symp. on the Theory of Computing*, pages 21–31, 1991.

[3] P. Beame, M. E. Saks, X. Sun, and E. Vee. Super-linear time-space tradeoff lower bounds for randomized computation. In *Proc. of the 41st Annu. IEEE Symp. on Foundations of Computer Science*, pages 169–179, 2000.

[4] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In C. Choffrut and T. Lengauer, editors, *STACS '90, 7th Annu. Symp. on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48. Springer-Verlag, 1990.

[5] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Locally random reductions: Improvements and applications. *J. of Cryptology*, 10(1):17–36, 1997. Early version: Security with small communication overhead, CRYPTO '90, volume 537 of *Lecture Notes in Computer Science*, pages 62-76. Springer-Verlag, 1991.

[6] A. Beimel and Y. Ishai. Information-theoretic private information retrieval: A unified construction. In P. G. Spirakis and J. van Leeuven, editors, *Proc. of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 912–926. Springer, 2001.

[7] A. Beimel, Y. Ishai, and E. Kushilevitz. Upper bounds for information-theoretic private information retrieval via a unified construction. Manuscript, 2002. This is a full version of [20] and [6].

[8] A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *Proc. of the 31st Annu. ACM Symp. on the Theory of Computing*, pages 89–98, 1999.

[9] A. Beimel, Y. Ishai, E. Kushilevitz, and J. F. Raymond. Breaking the $O(n^{\frac{1}{2k-1}})$ barrier for inforamtion-theoretic private information retrieval. In *Proc. of the 43rd Annu. IEEE Symp. on Foundations of Computer Science*, pages 261–270, 2002.

[10] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. In M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 56–74. Springer, 2000.

[11] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 1999.

[12] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of the 29th Annu. ACM Symp. on the Theory of Computing*, pages 304–313, 1997.

[13] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of the 36th Annu. IEEE Symp. on Foundations of Computer Science*, pages 41–51, 1995. Journal version: *J. of the ACM*, 45:965–981, 1998.

[14] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[15] G. Di-Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for private information retrieval. *J. of Cryptology*, 14(1):37–74, 2001. Preliminary version in *Proc. of the 17th Annu. ACM Symp. on Principles of Distributed Computing*, pages 91–100, 1998.

[16] G. Di-Crescenzo, T. Malkin, and R. Ostrovsky. Single-database private information retrieval implies oblivious transfer. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2000.

[17] Y. Dodis. Space-time tradeoffs for graph properties. Master's thesis, MIT, 1998.

[18] Y. Gertner, S. Goldwasser, and T. Malkin. A random server model for private information retrieval. In M. Luby, J. Rolim, and M. Serna, editors, *RANDOM '98, 2nd International Workshop on Randomization and Approximation Techniques in Computer Science*, volume 1518 of *Lecture Notes in Computer Science*, pages 200–217. Springer, 1998.

[19] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. of the 30th Annu. ACM Symp. on the Theory of Computing*, pages 151–160, 1998. Journal version: *J. of Computer and System Sciences*, 60(3):592–629, 2000.

[20] Y. Ishai and E. Kushilevitz. Improved upper bounds on information theoretic private information retrieval. In *Proc. of the 31st Annu. ACM Symp. on the Theory of Computing*, pages 79 – 88, 1999.

[21] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. Manuscript, 2002.

[22] T. Itoh. Efficient private information retrieval. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E82-A(1):11–20, 1999.

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

[24] E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for single-database computationally-private information retrieval. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 104–121, 2000.

[25] T. Malkin. *A Study of Secure Database Access and General Two-Party Computation.* PhD thesis, MIT, 2000. See: http://theory.lcs.mit.edu/∼cis/cis-theses.html.

[26] E. Mann. Private access to distributed information. Master's thesis, Technion – Israel Institute of Technology, Haifa, 1998.

[27] P. B. Miltersen. Cell probe complexity – a survey. *Advances in Data Structures* (Pre-conference Workshop of FSTTCS'99), 1999. See: http://www.daimi.au.dk/∼bromille/Papers/.

[28] P. B. Miltersen, N. Nisan, S. Safra, and A Wigderson. On data structures and asymmetric communication complexity. *J. of Computer and System Sciences*, 57(1):37–49, 1998.

[29] R. Ostrovsky and V. Shoup. Private information storage. In *Proc. of the 29th Annu. ACM Symp. on the Theory of Computing*, pages 294–303, 1997.

[30] J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 1998.

[31] A. Yamamura and T. Saito. Private information retrieval based on the subgroup membership problem. In V. Varadharajan and Y. Mu, editors, *ACISP 2001*, volume 2119 of *Lecture Notes in Computer Science*, pages 206–220. Springer-Verlag, 2001.

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