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

Amos Beimel[1], Yuval Ishai[2], and Tal Malkin[3]

[1] Dept. of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel. beimel@cs.bgu.ac.il.
[2] DIMACS and AT&T Labs – Research, USA. yuval@dimacs.rutgers.edu.
[3] AT&T Labs – Research, 180 Park Ave., Florham Park, NJ 07932, USA. tal@research.att.com.

Abstract. Private information retrieval (PIR) enables a user to retrieve a specific data item from a database, replicated among one or more servers, while hiding from each server the identity of the retrieved item. This problem was suggested by Chor et al. [11], 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 just one 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 save work. In particular, we construct, for any constant $k \geq 2$, a k-server protocol with $O(n^{1/(2k-1)})$ communication and $O(n/\log^{2k-2} n)$ work, and for any constants $k \geq 2$ and $\epsilon > 0$ a k-server protocol with $O(n^{1/k+\epsilon})$ communication and work. We also prove some lower bounds on the work of the servers when they are only allowed to store a small number of extra bits. Finally, we present some alternative approaches to saving computation, by batching queries or by moving most of the computation to an off-line stage.

1 Introduction

In this era of the Internet and www.bigbrother.com, it is essential to protect the privacy of the small user. An important aspect of this problem is hiding the information the 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. Towards this end, Chor, Goldreich, Kushilevitz, and Sudan [11] 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., [2, 25, 10, 20, 18, 17, 14, 9, 8, 19, 15, 21, 1]). 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) involved is at least n, the size of the entire database, for a single query of the user. 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 approaches 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.

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 [11]. To overcome this problem, Chor et al. [11] 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 to date are summarized below: (1) a 2-server protocol with communication complexity of $O\left(n^{1/3}\right)$ bits [11], (2) a k-server protocol, for a constant k, with communication complexity of $O\left(n^{1/(2k-1)}\right)$ bits [2] (improving on [11], see also [19]), and (3) a protocol with $O(\log n)$ servers and communication complexity of $O\left(\log^2 n \log\log n\right)$ bits [5, 6, 11]. In all these protocols it is assumed that the servers do not communicate with each other.[2]

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 proto-

[1] [17] 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.

[2] Extensions to t-private PIR protocols, in which the user is protected against collusions of up to t servers, have been considered in [11, 19, 7].

cols are called *computational* PIR protocols). Following a 2-server construction of Chor and Gilboa [10], Kushilevitz and Ostrovsky [20] proved that in this setting one server suffices; under a standard number theoretic assumption they construct, for every constant $\epsilon > 0$, a *single* server protocol with communication complexity of $O(n^\epsilon)$ bits. Cachin, Micali, and Stadler [9] present a single server protocol with polylogarithmic communication complexity, based on a new number theoretic intractability assumption. Other works in this setting are [25, 24, 27, 8, 15, 21, 1].

The only previous work that has addressed the servers' computation is that of Gertner, Goldwasser, and Malkin [17] (see also [23]), who 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 [14] present another model for PIR using special-purpose servers. By extending their technique, it is possible to shift most of the servers' work to an off-line stage, at the expense of requiring additional off-line work for each future query. This application is discussed in Section 5.

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 (or negligibly smaller than n in the computational setting). Consequently, we suggest the model of PIR with preprocessing: Before the first execution of the protocol each server computes and stores some 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.[3] The number of extra bits each server is allowed to store in the preprocessing stage is polynomial in n.

We demonstrate that preprocessing can save computation. There are three important performance measurements that we would like to minimize: communication, servers' work (i.e., computation), and storage. We describe a few protocols with different trade-offs between these parameters. We first construct, for any $\epsilon > 0$ and constant $k \geq 2$, a k-server protocol with $O(n^{1/(2k-1)})$ communication, $O\left(n/(\epsilon \log n)^{2k-2}\right)$ 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 communication compared to the best known information-theoretic PIR protocols. We define a combinatorial problem for which a better solution will further reduce the work in this protocol. 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. All the above protocols maintain information-theoretic user privacy.

[3] This problem can be rephrased in terms of Yao's cell-probe model [28]; in the full version of the paper we elaborate on the connection with this model.

We prove, on the negative side, that if the servers are only allowed to store a bounded number of bits in the preprocessing stage, then their computation in response to each query is big. In particular, we prove that if the servers are allowed to store only e extra bits ($e \geq 1$) in the preprocessing stage, then the expected work of the servers is $\Omega(n/e)$.

Finally, we suggest two alternative approaches for saving work. First, we suggest batching multiple queries to reduce the amortized work per query, and show how to achieve sub-linear work while maintaining the same communication. Second, we show how to shift most of the work to an off-line stage, applying a separate preprocessing procedure for each future query. While generally more restrictive than our default model, both of these alternative approaches may be applied in the single-server case as well.

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

2 Definitions

We first define one-round[4] information-theoretic PIR protocols. A k-server PIR protocol involves k servers $\mathcal{S}_1, \ldots, \mathcal{S}_k$, each holding the same n-bit string x (the database), and a user who wants to retrieve a bit x_i of the database.

Definition 1 (PIR). *A k-server PIR protocol $\mathcal{P} = (\mathcal{Q}_1, \ldots, \mathcal{Q}_k, \mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{C})$ consists of three types of algorithms: query algorithms $\mathcal{Q}_j(\cdot, \cdot)$, answering algorithms $\mathcal{A}_j(\cdot, \cdot)$, and a reconstruction algorithm $\mathcal{C}(\cdot, \cdot, \ldots, \cdot)$ ($\mathcal{C}$ has $k+2$ arguments). At the beginning of the protocol, the user picks a random string r and, for $j = 1, \ldots, k$, computes a query $q_j = \mathcal{Q}_j(i, r)$ and sends it to server $\mathcal{S}_j$. Each server responds with an answer $a_j = \mathcal{A}_j(q_j, x)$ (the answer is a function of the query and the database; without loss of generality, the servers are deterministic). Finally, the user computes the bit x_i by applying the reconstruction algorithm $\mathcal{C}(i, r, a_1, \ldots, a_k)$. A PIR protocol is secure if:*

Correctness. *The user always computes the correct value of x_i. Formally, $\mathcal{C}(i, r, \mathcal{A}_1(\mathcal{Q}_1(i, r), x), \ldots, \mathcal{A}_k(\mathcal{Q}_k(i, r), x)) = x_i$ for every $i \in \{1, ..., n\}$, every random string r, and every database $x \in \{0, 1\}^n$.*

Privacy. *Each server has no information about the bit that the user tries to retrieve: For every two indices i_1 and i_2, where $1 \leq i_1, i_2 \leq n$, and for every j, where $1 \leq j \leq k$, the distributions $\mathcal{Q}_j(i_1, \cdot)$ and $\mathcal{Q}_j(i_2, \cdot)$ are identical.*

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.

[4] All the protocols constructed in this paper, as well as most previous PIR protocols, are one-round. This definition may be extended to multi-round PIR in the natural way. All our results (specifically, our lower bounds) hold for the multi-round case.

Definition 2 (PIR with Preprocessing). *A PIR protocol with e* extra bits $\mathcal{P} = (\mathcal{E}, \mathcal{Q}_1, \ldots, \mathcal{Q}_k, \mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{C})$ *consists of* 4 *types of algorithms: preprocessing algorithm $\mathcal{E}$ which computes a mapping from $\{0,1\}^n$ to $\{0,1\}^e$, query and reconstruction algorithms $\mathcal{Q}_j$ and $\mathcal{C}$ which are the same as in regular PIR protocols, and the answer algorithms $\mathcal{A}_j(\cdot,\cdot,\cdot)$, which, in addition to the query q_j and the database x, have an extra parameter – the extra bits $\mathcal{E}(x)$. The privacy is as above and the correctness includes $\mathcal{E}$:*

Correctness with Extra Bits. *The user always computes the correct value of x_i. Formally, $\mathcal{C}(i, r, \mathcal{A}_1(\mathcal{Q}_1(i,r), x, \mathcal{E}(x)), \ldots, \mathcal{A}_k(\mathcal{Q}_k(i,r), x, \mathcal{E}(x))) = x_i$ for every $i \in \{1, ..., n\}$, every random string r, and every database $x \in \{0,1\}^n$.*

Next we define the work in a PIR protocol. We measure the work in a simplistic way, only counting the number of bits that the servers read (both from the database itself and from the extra bits). This 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, in all our protocols the servers' total work is linear in the number of bits they read.

Definition 3 (Work in PIR). *Fix a PIR protocol $\mathcal{P}$. For a query q and database $x \in \{0,1\}^n$ we denote the the number of bits that $\mathcal{S}_j$ reads from x and $\mathcal{E}(x)$ in response to q by* $\mathrm{BITS}_j(x,q)$.[5] *For a random string r of the user, 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,* $\mathrm{WORK}(i,x,r) \overset{def}{=} \sum_{j=1}^{k} \mathrm{BITS}_j(x, \mathcal{Q}_j(i,r))$. *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,* $\mathrm{WORK}(i,x) \overset{def}{=} \mathrm{E}_r\left[\mathrm{WORK}(i,x,r)\right]$.

Notation. We let $[m]$ denote the set $\{1, \ldots, m\}$. For a set A and an element i, define $A \oplus i$ as $A \cup \{i\}$ if $i \notin A$ and as $A \setminus \{i\}$ if $i \in A$. For a finite set A, 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. All logarithms are taken to the base 2. By H we denote the binary entropy function; that is, $H(p) = -p \log p - (1-p)\log(1-p)$.

3 Upper Bounds

We show that preprocessing can reduce the work. We start with a simple protocol which demonstrates ideas of the protocols in the rest of the section, in Section 3.1 we present a 2-server protocol with $O(n/\log^2 n)$ work, and in Section 3.2 we construct a k-server protocol, for a constant k, with $O(n/\log^{2k-2} n)$ work. In these protocols the communication is $O(n^{1/(2k-1)})$ for k servers. In Section 3.3 we describe a combinatorial problem concerning spanning of cubes; a good construction for this problem will reduce the work in the previous protocols. Finally, in Section 3.4 we utilize PIR protocols with short query complexity to obtain k-server protocols with $O\left(n^{1/k+\epsilon}\right)$ work and communication.

[5] Technically speaking, also $\mathcal{E}(x)$ should have been a parameter of BITS. However, since $\mathcal{E}$ is a function of x we can omit it.

A Warm-Up. We show that using $n^2/\log n$ extra bits we can reduce the work to $n/\log n$. 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. [11].

Original Protocol [11]. 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 a random subset A^j to S_j, and 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 $t = 1, \ldots, n/\log n$). Each server computes the exclusive-or for every subset of these sets. That is, for every t, where $1 \leq t \leq n/\log n$, and every $G \subseteq D_t$, each server computes and 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 $\bigoplus_{\ell \in A^j \cap D_1} x_\ell, \ldots, \bigoplus_{\ell \in A^j \cap D_{n/\log n}} x_\ell$.

3.1 A 2-Server Protocol with Improved Work

We describe, for every constant ϵ, a 2-server protocol with $O(n^{1+\epsilon})$ extra bits, $O\left(n^{1/3}\right)$ communication, and $O(n/(\epsilon^2 \log^2 n))$ work. Thus, our protocol exhibits tradeoff between the number of extra bits and the work. As the best known information-theoretic 2-server protocol without extra bits requires $O\left(n^{1/3}\right)$ communication, our protocol saves work without paying in the communication.

Theorem 1. *For every ϵ, 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 $O\left(n/(\epsilon^2 \log^2 n)\right)$ and the communication is $O(n^{1/3})$.*

Proof. We describe a simpler (and slightly improved) variant of a 2-server protocol of [11], and then show how preprocessing can save work for the servers.

Original Protocol (variant of [11]). 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 $\{0,1\}^{m^3}$ to $(\{0,1\}^m)^3$. In Fig. 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 $O(m) = O(n^{1/3})$.

Our Construction. To save on-line work the servers pre-compute the exclusive-or of some sub-cubes of bits. Let $\alpha = 0.5\epsilon \log n$. The set $[m]$ is partitioned to

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 $\ell \in [m]$

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

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

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

m/α disjoint sets $D_1, \ldots, D_{m/\alpha}$ of size α (e.g., $D_t = \{(t-1)\alpha + 1, \ldots, t\alpha\}$ for $t = 1, \ldots, m/\alpha$). For every $\ell \in [m]$, 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 $\bigoplus_{\ell_1 \in G_1, \ell_2 \in G_2} x_{\ell,\ell_1,\ell_2}$, $\bigoplus_{\ell_1 \in G_1, \ell_2 \in G_2} x_{\ell_1,\ell,\ell_2}$, and $\bigoplus_{\ell_1 \in G_1, \ell_2 \in G_2} x_{\ell_1,\ell_2,\ell}$. 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 $O(m^2/\alpha^2)$ bits. Thus, the number of bits that each server reads is $O(m^3/\alpha^2) = O\left(n/(\epsilon \log n)^2\right)$. $\square$

3.2 A k-Server Protocol with Small Communication

We present a k-server protocol with $O(n^{1+\epsilon})$ extra bits, $O\left(n^{1/(2k-1)}\right)$ communication, and $O\left(n/(\epsilon \log n)^{2k-2}\right)$ work for constant k. (The best known information-theoretic k-server protocol without extra bits requires the same communication and $O(n)$ work).

Theorem 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 $O\left((2k)^{4k} n/(\epsilon \log n)^{2k-2}\right)$ and the communication is $O(n^{1/(2k-1)})$. If k is constant, the work is $O\left(n/(\epsilon \log n)^{2k-2}\right)$, and if $k \leq 0.5(\log n)^{1/4}$ and $\epsilon \geq 1$ then the work is $O\left(n/(\epsilon \log n)^{k-2}\right)$.*

Proof. We save work in a k-server protocol of Ishai and Kushilevitz [19].

Original Protocol [19]. As the protocol of [19] involves some notation, we only describe its 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. A sub-cube is a $(d-1)$-dimensional sub-cube if there exists some r such that $|A_r| = 1$. In the protocol from [19] 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 $O\left(k^3 n^{1/(2k-1)}\right)$.

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 $\alpha = \frac{\epsilon \log n}{d-1}$. The set $[m]$ is partitioned to m/α disjoint sets $D_1, \ldots, D_{m/\alpha}$ of size α. For every $r \in \{1, \ldots, d\}$, every $\ell \in [m]$, every $t_1, t_2, \ldots, t_{d-1}$, where $1 \le t_1, t_2, \ldots, t_{d-1} \le m/\alpha$, every $G_1 \subseteq D_{t_1}$, every $G_2 \subseteq D_{t_2}$, ..., 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 $d^{d-1} < 2$ and since $\epsilon > 4k/\log n$). 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 $O\left((2k)^{4k} n/(\epsilon \log n)^{2k-2}\right)$. □

3.3 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 [16]. Our protocols in Section 3.1 and Section 3.2 are based on constructions for this problem; better constructions will enable to further reduce the work in these protocols.

We start with some notation. Consider the collection of all d-dimensional sub-cubes $\mathcal{F}_d \overset{\text{def}}{=} \{G_1 \times \ldots \times G_d \ : G_1, \ldots, G_d \subseteq [m]\}$. 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 ℓ is in an odd number of sets S_j.

Definition 4 (q-xor basis). $\mathcal{X} \subseteq 2^{[m]^d}$ *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, for $D_1, \ldots, D_{m/\log m}$ the partition of $[m]$, defined in Section 3.1, the collection $\mathcal{X}_0 \overset{\text{def}}{=} \{G_1 \times G_2 \ : \ \exists i, j \ G_1 \subseteq D_i, G_2 \subseteq D_j\}$ is a $m^2/\log^2 m$-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 1. *If $\mathcal{X}$ is a q-xor basis 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 [11], described in Fig. 1, in which $n = m^3$. For each set $S \in \mathcal{X}$, each server computes and stores $3m|\mathcal{X}|$ bits: for every $\ell \in [m]$ it stores the bits $\bigoplus_{(\ell_1,\ell_2) \in S} x_{\ell,\ell_1,\ell_2}$, $\bigoplus_{(\ell_1,\ell_2) \in S} x_{\ell_1,\ell,\ell_2}$, and $\bigoplus_{(\ell_1,\ell_2) \in S} x_{\ell_1,\ell_2,\ell}$. In the protocol of [11] 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 $O(mq) = O(n^{1/3}q)$.[6] $\square$

Lemma 1 suggests the following problem:

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

It can be shown that the smallest q for which there is a q-xor basis of $\mathcal{F}_d$ whose size is $\text{poly}(m^d)$ 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.

3.4 Utilizing PIR Protocols with Logarithmic Query Length

If we have a PIR protocol with logarithmic query length and 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 2. *If there is a k-server PIR protocol in which the length of the query sent to each server is α and the length of answer of each server is β, then there is a k-server PIR protocol with β work per server, $\alpha + \beta$ communication, and $2^\alpha \cdot \beta$ extra-bits.*

A 2-server PIR protocol with $\alpha = \log n$ and sub-linear β is implied by communication complexity results of [26, 4, 3]. The most recent of those, due to Ambainis and Lokam [3], implies an upper bound of $\beta = n^{0.728\ldots+o(1)}$.[7] We use similar techniques to construct a family of PIR protocols which provides a general tradeoff between α and β. In particular, our construction allows the exponent in the polynomial bounding the answer length to get arbitrarily close to 1/2 while maintaining $O(\log n)$ query length. At the heart of the construction is the following lemma of Babai, Kimmel, and Lokam [4]. Let $\Lambda(m, w) = \sum_{h=0}^{w} \binom{m}{h}$.

Lemma 3 ([4]). *Let $p(Y_1, Y_2, \ldots, Y_m)$ be a degree-d m-variate polynomial*[8] *over* GF(2)*. Let y_ℓ^h, where $1 \le h \le k$ and $1 \le \ell \le m$, be arbitrary km elements of* GF(2)*, and $y_\ell = \sum_{h=1}^{k} y_\ell^h$ for $\ell = 1, \ldots, m$. Suppose that each $\mathcal{S}_j$ knows all $(k-1)m$ bits y_ℓ^h with $h \ne j$ and the polynomial p, and that the user knows all km values y_ℓ^h but does not know p. Then, there exists a communication protocol in which each $\mathcal{S}_j$ simultaneously sends to the user a single message of length $\Lambda(m, \lfloor d/k \rfloor)$, and the user always outputs the correct bit value of $p(y_1, \ldots, y_m)$.*

The key idea in our construction is to apply Lemma 3 where $(y_1, y_2, \ldots, y_m)$ is a "convenient" encoding of the retrieval index i. Specifically, by using a low-weight encoding of i, the data bit x_i can be expressed as a low-degree polynomial

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

[7] This immediately implies a protocol with $n^{0.728\ldots+o(1)}$ communication and work and $n^{1.728\ldots+o(1)}$ extra-bits.

[8] A degree-d polynomial is a multi-linear polynomial of (total) degree *at most d*.

(depending on x) in the bits of the encoding. By letting the user secret-share the encoding of i among the servers in an appropriate manner, Lemma 3 will allow the servers to communicate x_i to the user efficiently. Low-weight encodings (over larger fields) have been previously used in PIR-related works [6, 11, 14]. However, it is the combination of this encoding with Lemma 3 which gives us the extra power.

Theorem 3. *Let m and d be positive integers such that $\Lambda(m,d) \geq n$. Then, for any $k \geq 2$, there exists a k-server PIR protocol with $\alpha = (k-1)m$ query bits and $\beta = \Lambda(m, \lfloor d/k \rfloor)$ answer bits per server.*

Proof. Assign a distinct length-m binary encoding $E(i)$ to each index $i \in [n]$, such that $E(i)$ contains at most d ones. (Such an encoding exists since $\Lambda(m,d) \geq n$.) For each $x \in \{0,1\}^n$, define a degree-d m-variate polynomial p_x over GF(2) such that $p_x(E(i)) = x_i$ for every $i \in [n]$.[9] Specifically, let $p_x(Y_1, \ldots, Y_m) = \sum_{i=1}^{n} x_i \cdot p^{(i)}(Y_1, \ldots, Y_m)$, where each $p^{(i)}$ is a fixed degree-d polynomial such that $p^{(i)}(E(i'))$ equals 1 if $i = i'$ and equals 0 if $i \neq i'$. (The polynomials $p^{(i)}$ can be constructed in a straightforward way; details are omitted from this version.) The protocol with the specified complexity is described below. The user encodes i as the m-bit string $y = E(i)$, and breaks each bit y_ℓ, where $1 \leq \ell \leq m$, into k additive shares $y_\ell^1, \ldots, y_\ell^k$; that is, $y_\ell^1, \ldots, y_\ell^{k-1}$ are chosen uniformly at random from GF(2), and y_ℓ^k is set so that the sum (i.e., exclusive-or) of the k shares is equal to y_ℓ. The user sends to each $\mathcal{S}_j$ the $(k-1)m$ shares y_ℓ^h with $h \neq j$. The query sent to each $\mathcal{S}_j$ consists of $(k-1)m$ uniformly random bits, guaranteeing the privacy of the protocol. By Lemma 3, each server can send $\Lambda(m, \lfloor d/k \rfloor)$ bits to the user such that the user can reconstruct $p_x(y_1, \ldots, y_m) = x_i$. □

We note that, by using constant-weight encodings, Theorem 3 can be used to improve the communication complexity of the 2-server protocol from [11] and its k-server generalizations from [2, 19] by constant factors. This and further applications of the technique are studied in [7]. For the current application, however, we will be most interested in denser encodings, in which the *relative* weight d/m is fixed as some constant θ, where $0 < \theta \leq 1/2$. In the following we rely on the approximation $2^{(H(\theta)-o(1))m} \leq \Lambda(m, \lfloor \theta m \rfloor) \leq 2^{H(\theta)m}$ (cf. [22, Theorem 1.4.5]). For $\Lambda(m, \lfloor \theta m \rfloor) \geq n$ to hold, it is sufficient to let $m = (1/H(\theta) + o(1)) \log n$. Substituting the above m and $d = \lfloor \theta m \rfloor$ in Theorem 3, and applying the transformation to PIR with preprocessing described in Lemma 2, we obtain:

Theorem 4. *For any integer $k \geq 2$ and constant $0 < \theta \leq 1/2$, there exists a k-server 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, since $H(\theta/k)/H(\theta)$ tends to $1/k$ as θ tends to 0, we have:

[9] The *existence* of an encoding $E : [n] \to \mathrm{GF}(2)^m$ such that x_i can be expressed as a degree-d polynomial in the encoding of i easily follows from the fact that the space of degree-d m-variate polynomials has dimension $\Lambda(m,d)$. We use the specific low-weight encoding for concreteness. Furthermore, the condition $\Lambda(m,d) \geq n$ is essential for the existence of such encoding.

Theorem 5. *For any constants $k \geq 2$ and $\epsilon > 0$ there exists a k-server protocol with polynomially many extra bits and $O\left(n^{1/k+\epsilon}\right)$ communication and work.*

The number of extra bits in the protocols of Theorem 4 may be quite large. By partitioning the database into small blocks, as in [11], it is possible to obtain a more general tradeoff between the storage and the communication and work. Specifically, by using blocks of size n^{μ}, where $0 < \mu \leq 1$, we obtain a 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.[10] It follows that for any constant $\epsilon > 0$ there exists a constant $\epsilon' > 0$ such that there is a 2-server protocol with $O(n^{1+\epsilon})$ extra bits and $O(n^{1-\epsilon'})$ communication and work.

Remark 1. There is a k-server PIR protocol with *one* extra bit (which is the exlusive-or of all bits in the database), $\frac{k}{2k-1} \cdot n$ work, and $O(n)$ communication. Thus, with 1 extra-bit we can save a constant fraction of the work. In Section 4 we show that with a constant number of bits at most a constant fraction of the computation can be saved, and if the bit is an exclusive-or of a subset of the data bits then the computation is at least $n/2$. Thus, this protocol illustrates that our lower bounds are essentially tight. The protocol will be described in the full version of this paper.

4 Lower Bounds

We prove that without preprocessing (namely without extra bits), the expected number of bits all servers must read 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.

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 the server has to read the entire database for each query, except with negligible probability: if the server does not read x_ℓ in response to some query, it knows that the user is not interested in x_ℓ, violating the user's privacy.

We start with some notation crucial for the lower bound. Fix a PIR protocol, and denote the user's random input by r. Let $\mathcal{C} \subseteq \{0,1\}^n$ be a set of strings (databases) to be fixed later. Define $B_j(i)$ as the set of all indices that server $\mathcal{S}_j$ reads in order to answer the user's query when the database is chosen uniformly

[10] In particular, the protocol obtained by letting $k = 2$, $\theta = 1 - 1/\sqrt{2}$, and $\mu = H(\theta)$ is very similar (and is slightly superior) to the protocol implied by [3].

[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.

from $\mathcal{C}$. Note that the set of bits $B_j(i)$ that $\mathcal{S}_j$ reads is a function of the query and the values of the bits that the server has already read. Since the query is a function of the index and the user's random input, the set $B_j(i)$ is a random variable of the user's random input r and the database $c \in_U \mathcal{C}$. Next define

$$\mathrm{P}(\ell) \overset{\mathrm{def}}{=} \max_{1 \leq i \leq n} \left\{ \Pr_{r,c} \left[\ell \in \bigcup_{j=1}^{k} B_j(i) \right] \right\}. \qquad (1)$$

That is, for every index i we consider the probability that at least one server reads x_ℓ on a query generated for index i, and $\mathrm{P}(\ell)$ is the maximum of these probabilities. Furthermore, define the random variable $B_j \overset{\mathrm{def}}{=} B_j(1)$ (by Lemma 4 below, the random variable B_j would not change if we choose another index instead of 1). Finally, for every ℓ define $\mathrm{P}_j(\ell) \overset{\mathrm{def}}{=} \Pr_{r,c}[\ell \in B_j]$, that is, the probability that x_ℓ is read by $\mathcal{S}_j$ (again, by Lemma 4 below, this probability is the same no matter which index i was used to generate the query).

4.1 Technical Lemmas

We start with three lemmas that will be used to establish our lower bounds. First note that, by the user's privacy, the view of $\mathcal{S}_j$, and in particular $B_j(i)$, is identically distributed for 1 and for any i. Thus,

Lemma 4. *For every $j \in \{1, \ldots, k\}$, every index $i \in [n]$, and every set $B \subseteq [n]$,* $\Pr_{r,c}[B_j(i) = B] = \Pr_{r,c}[B_j = B]$.

Lemma 5. *For every $j \in \{1, \ldots, k\}$ it hold that* $\mathrm{E}_{r,c}[|B_j|] = \sum_{\ell=1}^{n} \mathrm{P}_j(\ell)$.

Proof. Define the random variables $Y_1, \ldots, Y_n$ where $Y_\ell = 1$ if $\ell \in B_j$ and $Y_\ell = 0$ otherwise. Clearly, $\mathrm{E}_{r,c}[Y_\ell] = \Pr[Y_\ell = 1] = \mathrm{P}_j(\ell)$. Furthermore, $|B_j| = \sum_{\ell=1}^{n} Y_\ell$. Thus, $\mathrm{E}_{r,c}[|B_j|] = \mathrm{E}_{r,c}[\sum_{\ell=1}^{n} Y_\ell] = \sum_{\ell=1}^{n} \mathrm{E}_{r,c}[Y_\ell] = \sum_{\ell=1}^{n} \mathrm{P}_j(\ell)$. □

Next we prove that $\sum_{\ell=1}^{n} \mathrm{P}(\ell)$ is a lower bound on the expected number of bits the servers read, namely on the expected work for a random database in $\mathcal{C}$.

Lemma 6. *For every $i \in [n]$,* $\mathrm{E}_{r,c}\left[\sum_{j=1}^{k} |B_j(i)|\right] \geq \sum_{\ell=1}^{n} \mathrm{P}(\ell)$.

Proof. First, for an index $\ell \in [n]$ let i_ℓ be an index that maximizes the probability in the r.h.s. of (1), that is, $\mathrm{P}(\ell) = \Pr_{r,c}\left[\ell \in \bigcup_{j=1}^{n} B_j(i_\ell)\right]$. Second, by Lemma 4,

$$\mathrm{P}_j(\ell) = \Pr_{r,c}\left[\ell \in B_j(i_\ell)\right]. \qquad (2)$$

Therefore, using the union bound,

$$\mathrm{P}(\ell) = \Pr_{r,c}\left[\ell \in \bigcup_{j=1}^{n} B_j(i_\ell)\right] \leq \sum_{j=1}^{k} \Pr_{r,c}\left[\ell \in B_j(i_\ell)\right] = \sum_{j=1}^{k} \mathrm{P}_j(\ell). \qquad (3)$$

Third, by Lemma 4,

$$\mathrm{E}_{r,c}\left[|B_j(i)|\right] = \mathrm{E}_{r,c}\left[|B_j|\right]. \tag{4}$$

Thus, by linearity of the expectation, Equation (4), Lemma 5, and Inequality (3)

$$\begin{aligned}\mathrm{E}_{r,c}\left[\sum_{j=1}^{k}|B_j(i)|\right] &= \sum_{j=1}^{k}\mathrm{E}_{r,c}\left[|B_j(i)|\right] \;=\; \sum_{j=1}^{k}\mathrm{E}_{r,c}\left[|B_j|\right] \\ &= \sum_{j=1}^{k}\left(\sum_{\ell=1}^{n}\mathrm{P}_j(\ell)\right) \;=\; \sum_{\ell=1}^{n}\left(\sum_{j=1}^{k}\mathrm{P}_j(\ell)\right) \;\geq\; \sum_{\ell=1}^{n}\mathrm{P}(\ell). \qquad \square\end{aligned}$$

We express Lemma 6 as a lower bound on the work for a specific database.

Corollary 1. *For every PIR protocol there exists a database $c \in \{0,1\}^n$ such that for every $i \in [n]$, $\mathrm{WORK}(i,c) \geq \sum_{\ell=1}^{n}\mathrm{P}(\ell)$.*

Proof. By our definitions $\mathrm{E}_{r,c\in\mathcal{C}}\sum_{j=1}^{k}|B_j(i)| = \mathrm{E}_{r,c\in\mathcal{C}}\sum_{j=1}^{k}\mathrm{BITS}_j(c,\mathcal{Q}_j(i,r)) = \mathrm{E}_{c\in\mathcal{C}}\mathrm{WORK}(i,c)$. Thus, by Lemma 6, $\mathrm{E}_{c\in\mathcal{C}}\mathrm{WORK}(i,c) \geq \sum_{\ell=1}^{n}\mathrm{P}(\ell)$. Therefore, there must be some $c \in \mathcal{C}$ such that $\mathrm{WORK}(i,c) \geq \sum_{\ell=1}^{n}\mathrm{P}(\ell)$. $\square$

Remark 2. In the full version of this paper we prove that the corollary holds (up to a negligible difference) even if we replace the perfect privacy of the k-server PIR protocol with computational privacy. Thus, all the lower bounds in this section hold, up to a negligible difference, for k-server computational PIR protocols as well.

4.2 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. For simplicity we prove this lower bound for the case that the database is 0^n. The idea behind the lower bound is that one cannot obtain the value of x_ℓ without reading x_ℓ, thus for every query the user generates with index ℓ at least one server must read x_ℓ. This implies that $\mathrm{P}(\ell) = 1$ and the lower bound follows Corollary 1.

Theorem 6. *For every PIR protocol without extra bits and for every $i \in [n]$ $\mathrm{WORK}(i, 0^n) \geq n$.*

Proof. By Corollary 1 it is enough to prove that $\mathrm{P}(\ell) \geq 1$ for every ℓ. (Trivially, $\mathrm{P}(\ell) \leq 1$). Define $\mathcal{C} = \{0^n\}$, i.e., the probabilities $\mathrm{P}(\ell)$ are defined when the value of the database is 0^n. However, without reading the value of a bit, the servers do not know this value. If when the user queries about the ℓth bit no server reads this bit, then the answers of all the servers are the same for the databases 0^n and $0^{\ell-1}10^{n-\ell}$, so with this query for one of these databases the user errs with probability at least $1/2$ in the reconstruction of x_ℓ. Thus, by the correctness, for any possible query of the user generated with index ℓ, at least one of the servers must read x_ℓ. Thus, $\mathrm{P}(\ell) \geq \Pr_r\left[\ell \in \bigcup_{j=1}^{k} B_j(\ell)\right] = 1$. $\square$

4.3 Lower Bound with Extra Bits

In this section we show that a small number of extra bits cannot reduce the work too much. The proof uses information theory, and especially properties of the entropy function H (see, e.g., [12]).

To describe the ideas of the proof of the lower bound we first consider a special case where each of the e extra bits is an exclusive-or of a subset of the bits of the database. That is, there is a system of e linear equations over GF(2) that determines the values of the extra bits; the unknowns are the bits of the database. This is the case in all our protocols. (Better lower bounds for this case are presented in the end of this section.)

By Corollary 1 we need to prove that the probabilities $\mathrm{P}(\ell)$ are big. We fix the database to be $x = 0^n$, therefore the values of the extra bits are fixed to 0 as well. Assume towards a contradiction that $\mathrm{P}(\ell) < 1/(e+1)$ for at least $e+1$ indices, which, w.l.o.g., are $1, \ldots, e+1$. This implies that for every i, where $1 \le i \le e+1$, when the user is retrieving the ith bit, the servers, with positive probability, do not read any of the bits $x_1, \ldots, x_{e+1}$.

Now let $x_{e+2}, \ldots, x_n$ and all the extra bits be zero. We have established that in this case the servers with positive probability do not read the bits $x_1, \ldots, x_{e+1}$, and the user concludes that $x_1 = 0, \ldots, x_{e+1} = 0$ from the answers of the servers. Hence, by the correctness, it must hold that $x_1 = 0, \ldots, x_{e+1} = 0$. But in this case the linear system that determines the extra bits is reduced to a system of e homogeneous linear equations over GF(2) where the unknowns are the bits $x_1, \ldots, x_{e+1}$. Any homogeneous system with e equations and $e+1$ unknowns has a non-trivial solution. Therefore, there is a non-zero database in which $x_{e+2}, \ldots, x_n$ and all the extra bits be zero, contradiction since at least one bit x_i among $x_1, \ldots, x_{e+1}$ is not determined by $x_{e+2}, \ldots, x_n$ and the extra bits.

The above proof is only the rough idea of the proof of the general case. One problem in the general case is that we cannot fix the value of the database, and we need more sophisticated methods.

Theorem 7. *For every PIR protocol with e extra bits, there is some database $c \in \{0,1\}^n$ such that for every $i \in [n]$, $\mathrm{WORK}(i,c) \ge \frac{n}{4e} - \frac{1}{2}$.*

Proof. Since there are e extra bits, there exits a value for these bits that is computed for at least 2^{n-e} databases. Fix such a value for the extra bits and let $\mathcal{C}$ be the set of databases with this value for the extra bits. Thus, $|\mathcal{C}| \ge 2^{n-e}$. Let C be a random variable distributed uniformly over $\mathcal{C}$, and C_i be the ith bit of C. By definition,

$$\mathrm{H}(C) = \log |\mathcal{C}| \ge n - e. \tag{5}$$

We will prove that for all indices, but at most $2e$, it holds that $\mathrm{P}(\ell) \ge 1/(4e)$. Thus, the theorem follows from Corollary 1. Without loss of generality, assume that $\mathrm{P}(1) \le \mathrm{P}(2) \le \ldots \le \mathrm{P}(n)$. We start with a simple analysis of the entropies of C_i. First, by properties of conditional entropy,

$$\mathrm{H}(C) = \mathrm{H}(C_1 \ldots C_n) \le \mathrm{H}(C_{2e+1} \ldots C_n) + \sum_{\ell=1}^{2e} \mathrm{H}(C_\ell | C_{2e+1} \ldots C_n). \tag{6}$$

Second, since $C_{2e+1}C_{2e+2}\dots C_n$ obtains at most 2^{n-2e} values,

$$\mathrm{H}(C_{2e+1}C_{2e+2}\dots C_n) \le n-2e. \tag{7}$$

Combining (5), (6), and (7),

$$\sum_{\ell=1}^{2e} \mathrm{H}(C_\ell | C_{2e+1}C_{2e+2}\dots C_n) \ge \mathrm{H}(C_1\dots C_n) - \mathrm{H}(C_{2e+1}\dots C_n) \ge e. \tag{8}$$

The next lemma, together with (8), shows that not too many $\mathrm{P}(\ell)$ are small.

Lemma 7. *If* $\mathrm{P}(\ell) < \frac{1}{4e}$ *for every* $\ell \in [2e]$, *then* $\mathrm{H}(C_\ell | C_{2e+1}\dots C_n) < 0.5$ *for every* $\ell \in [2e]$.

Proof. Fix ℓ and consider an experiment where the database c is chosen uniformly from $\mathcal{C}$ and the PIR protocol is executed with the user generating a random query for index ℓ. With probability at least half, none of the bits $x_1, \dots, x_{2e}$ are read by any server in this execution (the probability is taken over the random input of the user and over the uniform distribution of $c \in \mathcal{C}$). Denote by $\mathcal{C}' \subseteq \mathcal{C}$ the set of all strings in $\mathcal{C}$ for which there is a positive probability that none of the bits $x_1, \dots, x_{2e}$ is read by any server (this time the probability is taken only over the random input of the user). Thus, $|\mathcal{C}'| \ge 0.5|\mathcal{C}|$. Since the user always reconstructs the correct value of the bit x_ℓ, then for every $c' \in \mathcal{C}'$ the values of the bits $c'_{2e}, \dots, c'_n$ determine the value of c'_ℓ; that is, for every $c \in \mathcal{C}$ if $c_m = c'_m$ for every $m \in \{2e+1, \dots, n\}$, then $c_\ell = c'_\ell$. Now, define a random variable Z where $Z = 1$ if the values of the bits $c_{2e}, \dots, c_n$ determine the value of c_ℓ and $Z = 0$ otherwise. In particular, Z must be 1 for any string in $\mathcal{C}'$. Hence,

$$\Pr_c[Z=1] \ge 0.5. \tag{9}$$

Furthermore,

$$\mathrm{H}(C_\ell | C_{2e+1}\dots C_n Z = 1) = 0. \tag{10}$$

On the other hand, since C_ℓ obtains at most two values,

$$\mathrm{H}(C_\ell | C_{2e+1}\dots C_n Z = 0) \le \mathrm{H}(C_\ell) \le 1. \tag{11}$$

By definition of conditional entropy, (11), (10), and (9)

$$\begin{aligned} &\mathrm{H}(C_\ell | C_{2e+1}\dots C_n Z) \\ &= \Pr[Z=0]\cdot \mathrm{H}(C_\ell | C_{2e+1}\dots C_n Z = 0) + \Pr[Z=1]\cdot \mathrm{H}(C_\ell | C_{2e+1}\dots C_n Z = 1) \\ &\le \Pr[Z=0] \ < \ 0.5. \end{aligned} \tag{12}$$

The values of $C_{2e+1}, \dots, C_n$ determine the value of Z, i.e., $\mathrm{H}(Z | C_{2e+1}\dots C_n) = 0$. Thus, $\mathrm{H}(C_\ell | C_{2e+1}\dots C_n) = \mathrm{H}(C_\ell | C_{2e+1}\dots C_n Z) < 0.5$. □

Lemma 7 and (8) imply that $\mathrm{P}(\ell) \ge 1/4e$ for at least one $\ell \in \{1, \dots, 2e\}$. Since we assume, without loss of generality, that $\mathrm{P}(1) \le \mathrm{P}(2) \le \dots \le \mathrm{P}(n)$, then $\sum_{\ell=1}^{n} \mathrm{P}(\ell) \ge \sum_{\ell=2e+1}^{n} \mathrm{P}(\ell) \ge (n-2e)/4e$, and by Corollary 1 the work of servers is as claimed in the theorem. □

Better Lower Bounds for Exclusive-or Extra Bits. If each extra bit is an exclusive-or of the bits of a subset of the database, then the lower bound of Theorem 7 can be improved by a factor of $\log n$, as stated in the following theorem (whose proof is omitted).

Theorem 8. *If $e < \log n$, then in every k-server PIR protocol with e exclusive-or extra bits the work of the servers is at least $(n - 2^e)/2$. If $\log n \leq e \leq \sqrt{n}$, then in every k-server PIR protocol with e exclusive-or extra bits the work is $\Omega(n \log n/e)$.*

As explained in Remark 1, the lower bound for a constant number of extra bits is essentially tight, as a matching upper bound protocol with one extra bit exists.

5 Alternative Approaches 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 approaches 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, which is an important advantage over the solutions of Section 3.

5.1 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.[12] 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.[13]

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 Fig. 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 $a^t X b$,

[12] This is optimal by the lower bound of Theorem 6.

[13] In this setting, the amortized communication complexity cannot be smaller than the communication complexity of a corresponding (single-query) PIR protocol.

where a and b are vectors in $\mathrm{GF}(2)^{n^{1/3}}$ determined by the user's query, and X 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 $A \cdot X \cdot B$, 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 ω denote the exponent of matrix multiplication (Coppersmith and Winograd [13] 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 $n^{1/3}$.

Finally, we note that the same approach can also be employed towards reducing the amortized work in computational single-server PIR protocols, when batching queries of users who share the same key. In the protocols from [20, 24, 27], which utilize homomorphic encryption, 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. It follows (e.g., by modifying the protocol from [20]) that for any constant $\epsilon > 0$ there is a constant $\epsilon' > 0$, such that there exists a single-server PIR protocol with $O(n^\epsilon)$ batch size, $O(n^\epsilon)$ communication, $O(n^{1-\epsilon'})$ amortized work, and sub-linear extra space.

5.2 Off-Line Interaction

In the PIR with preprocessing model, a single off-line computational effort 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 and "reasonable" off-line work.[14]

Towards achieving the above goal we extend an idea from [14]. Given any k-server PIR protocol ($k \geq 1$) in which the user sends α query bits to each server and receives β bits in return, Di-Crescenzo et al. [14] show how to construct another k-server protocol where: (1) in the off-line stage the user sends α bits

[14] This may be compared to the approach of Gertner et al. [17], where instead of shifting most computation to a "more reasonable place" (special purpose servers), here we shift most computation to a "more reasonable time" (the off-line stage).

to each server and receives nothing in return; (2) in the on-line stage the user sends $\log n$ bits to each server and receives β bits in return. 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, with βn storage, $O(\alpha)$ off-line communication and polynomial off-line computation, the on-line work is reduced to $O(\beta)$. Fortunately, most known PIR protocols admit variants in which the answer complexity β is very small, as small as a single bit in the multi-server case, while α is still sub-linear (see [14] for a detailed account). For instance, in the 2-server computational PIR protocol of Chor and Gilboa [10], $\alpha = 2^{O\left(\sqrt{\log n}\right)}$ and $\beta = 1$. Furthermore, by utilizing the structure of specific PIR protocols (including the one from [10]), the off-line computation of each server may be reduced to multiplying a length-n data vector by an $n \times n$ Toeplitz matrix determined by the user's query. Thus, using the FFT algorithm, the total off-line computation can be made very close to linear.

6 Open Problems

We have shown that using preprocessing in PIR protocols one can obtain polynomial 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 further research. The obvious open problem is if more substantial savings are possible:

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 5) are multi-server, information-theoretic 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. Part of the work of Amos Beimel was done while in Harvard University, supported by grants ONR-N00014-96-1-0550 and ARO-DAAL03-92-G0115.

References

1. W. Aiello, S. Bhatt, R. Ostrovsky, and S. Rajagopalan. Fast Verification of Any Remote Procedure Call: Short Witness-Indistinguishable One-Round Proofs for NP. In *ICALP 2000*.
2. A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *24th ICALP*, volume 1256 of *LNCS*, pages 401–407, 1997.
3. A. Ambainis and S. Lokam. Improved upper bounds on the simultaneous messages complexity of the generalized addressing function. In *LATIN 2000*.
4. L. Babai, P. Kimmel, and S. Lokam. Simultaneous messages vs. communication. In *12th STACS*, volume 900 of *LNCS*, pages 361–372, 1995.

5. D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *7th STACS*, volume 415 of *LNCS*, pages 37–48. Springer-Verlag, 1990.
6. D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Locally random reductions: Improvements and applications. *J. of Cryptology*, 10:17–36, 1997. Early version: Security with small communication overhead, *CRYPTO '90*.
7. A. Beimel and Y. Ishai. On private information retrieval and low-degree polynomials. Manuscript, 2000.
8. A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *31th STOC*, pages 89–98, 1999.
9. C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT '99*, volume 1592 of *LNCS*, pages 402–414. Springer, 1999.
10. B. Chor and N. Gilboa. Computationally private information retrieval. In *29th STOC*, pages 304–313, 1997.
11. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *36th FOCS*, pages 41–51, 1995. Journal version: *JACM*, 45:965–981, 1998.
12. T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.
13. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9:251-280, 1990.
14. G. Di-Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for database private information retrieval. In *17th PODC*, pages 91–100, 1998.
15. G. Di-Crescenzo, T. Malkin, and R. Ostrovsky. Single-database private information retrieval implies oblivious transfer. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 122 –138, 2000.
16. Y. Dodis. Space-Time Tradeoffs for Graph Properties. Master's thesis, Massachusetts Institute of Technology, 1998.
17. Y. Gertner, S. Goldwasser, and T. Malkin. A random server model for private information retrieval. In *RANDOM '98, 2nd Workshop on Randomization and Approximation Techniques in CS*, vol. 1518 of *LNCS*, pages 200–217. 1998.
18. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *30th STOC*, pages 151–160, 1998.
19. Y. Ishai and E. Kushilevitz. Improved upper bounds on information theoretic private information retrieval. In *31th STOC*, pages 79 – 88, 1999.
20. E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *38th FOCS*, pages 364–373, 1997.
21. E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 104–121, 2000.
22. J. H. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1982.
23. T. Malkin. *A Study of Secure Database Access and General Two-Party Computation.* PhD thesis, MIT, 2000. http://theory.lcs.mit.edu/∼cis/cis-theses.html .
24. E. Mann. Private access to distributed information. Master's thesis, Technion - Israel Institute of Technology, Haifa, 1998.
25. R. Ostrovsky and V. Shoup. Private information storage. In *29th STOC*, pages 294–303, 1997.
26. P. Pudlák and V. Rödl. Modified Ranks of Tensors and the Size of Circuits. In *25th STOC*, pages 523–531, 1993.
27. J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *ASIACRYPT '98*, volume 1514 of *LNCS*, pages 357–371. Springer, 1998.
28. A.C. Yao. Should tables be sorted? *JACM*, 28:615–628, 1981.