

Breaking the $O(n^{1/(2^k-1)})$ Barrier for Information-Theoretic Private Information Retrieval*

Amos Beimel[†] Yuval Ishai[‡] Eyal Kushilevitz[§] Jean-François Raymond[¶]

April 24, 2006

Abstract

Private Information Retrieval (PIR) protocols allow a user to retrieve a data item from a database while hiding the identity of the item being retrieved. Specifically, in *information-theoretic k -server* PIR protocols the database is replicated among k servers, and each server learns nothing about the item the user retrieves. The cost of such protocols is measured by the *communication complexity* of retrieving one out of n bits of data. For any fixed k , the complexity of the best protocols prior to our work was $O(n^{\frac{1}{2^k-1}})$ (Ambainis, 1997). Since then several methods were developed in an attempt to beat this bound, but all these methods yielded the same asymptotic bound.

In this work, this barrier is finally broken and the complexity of information-theoretic k -server PIR is improved to $n^{O(\frac{\log \log k}{k \log k})}$. The new PIR protocols can also be used to construct k -query binary *locally decodable codes* of length $\exp(n^{O(\frac{\log \log k}{k \log k})})$, compared to $\exp(n^{\frac{1}{k-1}})$ in previous constructions. The improvements presented in this paper apply even for small values of k : the PIR protocols are more efficient than previous ones for every $k \geq 3$, and the locally decodable codes are shorter for every $k \geq 4$.

1 Introduction

A Private Information Retrieval (PIR) protocol allows a user to retrieve a data item of its choice from a database while preventing the server storing the database from gaining information about the identity of the chosen item. This problem was introduced by Chor, Goldreich, Kushilevitz, and Sudan [17] and has since attracted a considerable amount of attention. In formalizing the problem, it is convenient to model the database by an n -bit string x where the user, holding some *retrieval index* i , wishes to learn the i -th data bit x_i . A trivial solution to the PIR problem is to send the entire database x to the user. However, while being perfectly private, the *communication complexity* of this solution may be prohibitively large. Indeed, the most significant goal of PIR-related research has been to minimize the communication complexity of PIR protocols. Unfortunately, if the server is not allowed to gain *any* information about the identity of the retrieved bit, then the linear communication complexity of the trivial solution is optimal [17]. To get around

* A preliminary version of this paper appeared in [9]. Some preliminary results appeared in [42].

[†] CS Dept., Ben-Gurion University, Beer-Sheva 84105, Israel. E-mail: beimel@cs.bgu.ac.il.

[‡] CS Dept., Technion, Haifa 32000, Israel. E-mail: yuvali@cs.technion.ac.il. Work done in part while at Princeton University, AT&T Labs – Research, and DIMACS. Supported in part by grant 36/03 from the Israel Science Foundation.

[§] CS Dept., Technion, Haifa 32000, Israel. E-mail: eyalk@cs.technion.ac.il. Supported by a grant from the Mitchell Schoref Fund and by grant 36/03 from the Israel Science Foundation.

[¶] CS Dept., Harvard University. E-mail: jraymond@eecs.harvard.edu. Work done in part while at McGill University's School of CS. Partly funded by a NSERC PGS-A scholarship.

this problem, Chor et al. [17] suggested that the user access k replicated copies of the database stored at different servers, requiring that each individual server get absolutely no information about i . PIR in this setting is called *information-theoretic* PIR.¹

Prior to the current work, the communication complexity of the best known information-theoretic PIR protocols was $O(n^{1/(2k-1)})$. This was first obtained for $k = 2$ servers in [17] and generalized to any fixed value of k by Ambainis [1]. This upper bound remained the best known until this work, in spite of various attempts to improve it [30, 31, 7] (see also [8, 47]). While these attempts resulted in finding new, very different, PIR protocols, they all ended up with the same $O(n^{1/(2k-1)})$ bound. (The constants, which depend on k , were significantly improved; this is in addition to asymptotic improvements for some extensions of the basic problem.) Note that the number of servers, k , is usually considered to be “small” and, in particular, independent of the length of the database, n ; for larger values of k , there is a construction ([17] and implicitly in [4, 5]) that gives an $O(\log n)$ -server protocol with $O(\log^2 n \log \log n)$ communication bits or alternatively an $O(\log n / \log \log n)$ -server protocol with $\text{polylog}(n)$ communication.

Other than the interest in PIR protocols for their own sake, they also found various applications. For example, PIR can be used as a building block for obtaining sublinear-communication protocols for more general problems of secure computation (e.g., [16, 22, 38, 13]). A more surprising application of PIR is to the construction of *locally decodable* error-correcting codes. A k -query Locally Decodable Code (LDC) allows to encode a database $x \in \{0, 1\}^n$ into a string y of length m , such that even if a large portion of y is adversarially corrupted, each bit of x can still be decoded *with high probability* by probing k , randomly selected, symbols of y . (See Section 4 for a more precise definition, and [45, 28] for surveys containing results on LDC.) Katz and Trevisan [33] have shown a close relation between LDC and information-theoretic PIR. In particular, any information-theoretic k -server PIR protocol (with a single round of queries and answers) can be transformed into a k -query LDC whose length is exponential in the length of the PIR queries and whose alphabet size is exponential in the length of the PIR answers. The best previously known upper bound on the length of a k -query *binary* LDC was $m(n) = 2^{O(n^{1/(k-1)})}$. This bound was obtained from PIR protocols with queries of length $O(n^{1/(k-1)})$ and a single answer bit per server.

1.1 Our results

We improve over the previous upper bounds for information-theoretic PIR and LDC. Our main contribution is a k -server PIR protocol whose communication complexity is $O(n^{\frac{c \log \log k}{k \log k}})$, for some constant c . (More specifically, our analysis shows that $c = 2$ can be used for every $k \geq 3$.) This protocol can be transformed in a *generic* way [27, 33] into a k -query binary LDC of length $\exp(n^{\frac{c' \log \log k}{k \log k}})$. However, we also provide a direct construction which is significantly better for small values of k . Our protocol is recursive and its analysis is obtained via the solution of a certain recurrence. As mentioned, the most interesting values of k are small ones. Hence, for several such values, we present in Figure 1 an analysis of the communication complexity where the exponent is determined exactly. The results in this figure show that our bounds are better than the previous ones for values which are as small as $k = 3$ in the case of PIR and $k = 4$ in the case of binary LDC.

An interesting difference between the binary LDC obtained in this work and the previous best LDC is that the latter enjoy the additional property of *self-correction*. That is, previous LDC allow to recover not only every bit of the database x but also every bit of its *encoding* y . The improved LDC obtained in this work are not known to support self-correction.

¹The term “information-theoretic PIR” may more generally refer to protocols which leak a small amount of information about i . However, there is no evidence that such a relaxation can yield significantly better protocols.

k	communication of k -server PIR		length of k -query binary LDC	
	previous	new	previous	new
2	$O(n^{1/3})$	–	$2^{O(n)}$	–
3	$O(n^{1/5})$	$O(n^{4/21}) = O(n^{1/5.25})$	$2^{O(n^{1/2})}$	–
4	$O(n^{1/7})$	$O(n^{8/63}) = O(n^{1/7.87})$	$2^{O(n^{1/3})}$	$2^{O(n^{3/10})}$
5	$O(n^{1/9})$	$O(n^{64/693}) = O(n^{1/10.82})$	$2^{O(n^{1/4})}$	$2^{O(n^{1/5})}$
6	$O(n^{1/11})$	$O(n^{32/441}) = O(n^{1/13.78})$	$2^{O(n^{1/5})}$	$2^{O(n^{1/7})}$

Figure 1: Upper bounds for small values of k .

1.2 Techniques

Our construction borrows some ideas from previous related works. These include the idea of representing the database using polynomials (as in [17, 3] and especially [8]), the use of replication-based secret sharing and the notion of “blocks” from [8], and the idea of recursively retrieving bits from the servers’ answers (instead of sending the whole answers) as a way to reduce communication. Recursion was used previously in PIR protocols [1, 15, 35]; however, our recursion is somewhat more sophisticated. Assume that we have a PIR protocol \mathcal{P} with the following three properties:

- The queries are short, however, the answers are long.
- The user only needs few bits from each answer.²
- There is an overlap between the answers that different servers send to the user. More precisely, each answer consists of several sub-answers and each sub-answer is known to several servers.

This protocol leads to a recursive protocol \mathcal{P}' as follows: The user sends its (short) queries as in \mathcal{P} , and each server computes its answer. However, the servers do not send their long answers to the user; instead the user and each subset of servers that hold a common sub-answer execute a PIR protocol in which the user retrieves the bits it needs from this sub-answer. The difficulty of constructing an appropriate protocol \mathcal{P} , to be used in the recursion, is in the somewhat contradicting goals of the above description. On one hand, we want the number of sub-answers and their size to be as small as possible. On the other hand, we want the “replication” (i.e., the overlap between sub-answers) to be as large as possible. Organizing the answers appropriately into sub-answers with good parameters according to the paradigm suggested above is not straightforward. Most of the technical work in this paper shows, in a sense, how to construct a protocol \mathcal{P} with such properties. We refer the reader to Appendix A for a more detailed high level description of our main protocol, explaining the source for improvement over previous approaches.

1.3 Related work

Several extensions of the basic PIR model were studied. These include extensions to t -private protocols, in which the user is protected against collusions of up to t servers [17, 8, 47]; extensions which protect the servers holding the database (in addition to the user), termed symmetric PIR (SPIR) [26, 39]; and other

²The user cannot reveal to the servers which bits it needs since this information might disclose the index i it is interested in.

extensions [41, 25, 20, 10, 13, 11]. PIR was also studied in a *computational* setting where privacy should only hold against computationally bounded servers; computational PIR was studied in both the multi-server setting [15] and a single server setting [35, 44, 37, 12, 14, 36, 24]. In contrast to information-theoretic PIR, computational PIR protocols with sublinear communication exist even in the single-server case (under standard cryptographic assumptions).

From a practical point of view, single-server PIR protocols are preferable to multi-server ones for obvious reasons: they avoid the need to maintain replicated copies of the database or to compromise the user’s privacy against several colluding servers. Moreover, single-server protocols from the literature obtain better asymptotic communication complexity than information-theoretic protocols with a constant number of servers. However, for typical real-life parameters the known single-server protocols are less efficient than known multi-server (even 2-server) protocols due to their high computational overhead. Furthermore, single-server protocols have some *inherent* limitations which can only be avoided in a multi-server setting. For instance, it is impossible for a (sublinear-communication) single-server PIR protocol to have very short queries (say, $O(\log n)$ bits long) sent from the user to the server, or very short answers (say, one bit long) sent in return. These two extreme types of protocols, which can be realized in the information-theoretic setting, have various applications [20, 10]. Finally, the close relation between information-theoretic PIR and locally decodable codes [33] further motivates the study of PIR in this setting.

No strong general lower bounds on PIR are known. Mann [37] obtained a constant-factor improvement over the trivial $\log_2 n$ bound, for any constant k . The constant was subsequently improved in [46]. In the 2-server case, much stronger lower bounds can be shown under the restriction that the user reconstructs x_i by reading a *constant* number of bits from the answers sent by the servers. Such bounds were first obtained for the case of *linear* protocols (where each query defines a linear function of the database) [29], and subsequently generalized to arbitrary protocols [34, 46]. Other lower bounds for restricted PIR protocols appear in [32, 6] and very recently in [43] under restrictions that apply to all known upper bounds including those in our paper. Lower bounds for locally decodable codes appear in [33, 19, 29, 40, 34, 46, 21]. These results still leave an exponential gap between known upper bounds and lower bounds in the general (unrestricted) case.

The PIR problem was studied in many different settings other than the basic ones discussed above. The reader is referred to a survey by Gasarch [23] for an overview of different flavors of PIR. Information-theoretic PIR in a *quantum* model was considered in [34]. In particular, [34] present a quantum 2-server protocol with $O(n^{3/10})$ complexity, improving over the $O(n^{1/3})$ complexity of the best known classical protocol. Interestingly, the quantum protocol relies on an improved 4-server (classical) protocol with one-bit answers constructed in the current work (see Section 4).

Subsequent work. Woodruff and Yekhanin [47] have recently suggested an elegant “geometric” approach to information-theoretic PIR. Using their approach, they obtain several improvements over previous protocols, including an improvement over the t -private protocol from [8] by roughly a factor of $\binom{k}{t}$. The asymptotic complexity of the protocol presented in this work is not improved by [47], and in fact is only matched for $k \leq 25$.

Organization. Following some definitions (Section 2), we present our main protocol in Section 3. In Section 4, we describe PIR protocols with short answers and their applications to locally-decodable codes. In Section 5, we describe an abstract framework which generalizes the concrete protocol presented in Section 3, and provides an alternative derivation of the main result. In Appendix A, we give a high-level description of our main protocol, explaining the source for improvement over previous approaches. Finally, in Appendix B,

we show that the choice of parameters in our protocol is essentially optimal, and that the generalization provided by the abstract framework from Section 5 only leaves a modest room for improvement.

2 Preliminaries

We use in our protocols multivariate polynomials. By default, all polynomials are over $\text{GF}(2)$. Variables of such polynomials are denoted with capital letters, e.g., Z_h ; assignments to these variables are in small letters, e.g., z_h . The term *degree- d* polynomial refers to a multivariate polynomial whose *total* degree is at most d . For an integer t , $[t]$ denotes the set $\{1, \dots, t\}$. Finally, $\log r$ should be read as $\log_2 r$.

A k -server PIR protocol involves k servers $\mathcal{S}_1, \dots, \mathcal{S}_k$, each holding the same n -bit string x (the database), and a user \mathcal{U} who knows n and wants to retrieve some bit x_i , where $i \in [n]$, without revealing i . We restrict our attention to *one-round*, 1-private, information-theoretic PIR protocols.

Definition 2.1 (PIR) *A PIR protocol is a triplet of algorithms $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$. At the beginning of the protocol, the user \mathcal{U} invokes $\mathcal{Q}(k, n, i)$ to pick a (randomized) k -tuple of queries (q_1, q_2, \dots, q_k) , along with an auxiliary information string aux . It sends each server \mathcal{S}_j the query q_j and keeps aux for a later use. Each server \mathcal{S}_j responds with an answer $a_j = \mathcal{A}(k, j, x, q_j)$. (We can assume without loss of generality that the servers are deterministic; hence, each answer is a function of the query and the database.) Finally, \mathcal{U} computes its output by applying the reconstruction algorithm $\mathcal{C}(k, n, a_1, \dots, a_k, \text{aux})$. We view the number of servers k as constant, and require all algorithms to be efficient in the data length n . A protocol \mathcal{P} restricted to a fixed k will be referred to as a k -server protocol. A protocol as above should satisfy the following requirements:*

Correctness. *For any $k, n, x \in \{0, 1\}^n$ and $i \in [n]$, the user outputs the correct value of x_i with probability 1 (where the probability is over the randomness of \mathcal{Q}).*

Privacy. *Each server learns no information about i . Formally, for any $k, n, i_1, i_2 \in [n]$, and server $j \in [k]$, the distributions $\mathcal{Q}_j(k, n, i_1)$ and $\mathcal{Q}_j(k, n, i_2)$ are identical, where \mathcal{Q}_j denotes the j -th output of \mathcal{Q} .*

The *communication complexity* of a PIR protocol \mathcal{P} , denoted $C_{\mathcal{P}}(n, k)$, is a function of k and n measuring the *total* number of bits communicated between the user \mathcal{U} and the k servers maximized over all choices of $x \in \{0, 1\}^n$, $i \in [n]$, and random inputs. The *query length* of \mathcal{P} , denoted $Q_{\mathcal{P}}(n, k)$, is the maximal number of bits sent from \mathcal{U} to *any* single server, and the *answer length*, denoted $A_{\mathcal{P}}(n, k)$, is the maximal number of answer bits sent by *any* server.

Finally, we say that a PIR protocol \mathcal{P} is *linear* if $\mathcal{A}(k, j, x, q)$ is a linear function of x for any fixed k, j, q (where linearity is over $\mathbb{F} = \text{GF}(2)$ by default). Note that in a linear PIR protocol \mathcal{U} can recover x_i by taking the exclusive-or of some subset of the answer bits determined by its queries. All protocols constructed in this work are linear.

3 Main Protocol

In this section, we present a PIR protocol that achieves the desired upper bound. It builds upon several ideas that are borrowed from [30, 7, 8]; however, for self containment, the presentation assumes no knowledge of these works.

3.1 The Framework

The protocol is based on representing the n -bit database x by a multivariate polynomial $P_x(Z_1, \dots, Z_m)$ over $\text{GF}(2)$. In this representation we carefully control two parameters: the degree d and the number of variables m which is chosen such that $m = \Theta(n^{1/d})$.³ The polynomial P_x should represent x in the following sense: with every $i \in [n]$ we associate a distinct assignment (also referred to as “encoding”) $E(i) \in \{0, 1\}^m$ such that the polynomial P_x satisfies

$$\forall i \in [n], \quad P_x(E(i)) = x_i. \quad (1)$$

(We do not care about the value $P_x(\vec{z})$ for assignments \vec{z} which are not of the form $E(i)$, for some i .)

To meet the above goals, we construct E and P_x as follows. Let $E(1), \dots, E(n)$ be n distinct binary vectors (strings) of length m and weight d . Such vectors exist if $\binom{m}{d} \geq n$, i.e., $m = \Theta(n^{1/d})$ variables are indeed sufficient.⁴ Define

$$P_x(Z_1, \dots, Z_m) \stackrel{\text{def}}{=} \sum_{i=1}^n x_i \prod_{\ell: E(i)_\ell=1} Z_\ell,$$

where $E(i)_\ell$ is the ℓ th bit of $E(i)$. Since each $E(i)$ is of weight d , the degree of P_x is at most d . Each assignment $E(i)$ to the variables Z_1, \dots, Z_m satisfies exactly one monomial in P_x (whose coefficient is x_i); thus, $P_x(E(i)) = x_i$.

Note that P_x is determined by x and is thus known to all servers. The user \mathcal{U} , on the other hand, does not know x . It has an index i , pointing to the bit from x it is interested in, and it can compute $E(i)$. Hence, the PIR problem is reduced to the problem of evaluating $P_x(E(i))$ while keeping $E(i)$ secret from each server. To this end, \mathcal{U} chooses at random points $\vec{y}_1, \dots, \vec{y}_k \in \{0, 1\}^m$ subject to the constraint

$$E(i) = \sum_{j=1}^k \vec{y}_j \quad (2)$$

and sends to each server \mathcal{S}_j all the \vec{y} 's except \vec{y}_j (i.e., $k - 1$ points to each server). Note that since each $k - 1$ of the \vec{y} 's are uniformly and independently distributed, a single server can learn no information about i . The user's goal is to evaluate $P_x(\sum_{j=1}^k \vec{y}_j) = P_x(E(i)) = x_i$. (Each point \vec{y}_j consists of m values $y_{j,h}$, for $j \in [k]$ and $h \in [m]$.) Equivalently, we can think of each variable Z_h of P_x as the sum of k variables: $Z_h = \sum_{j=1}^k Y_{j,h}$. The value $P_x(E(i))$ is obtained by assigning the value $y_{j,h}$ to each variable $Y_{j,h}$. Let Q_x be the polynomial obtained by viewing P_x as a polynomial in the variables $\{Y_{j,h}\}_{j \in [k], h \in [m]}$. This is a degree- d polynomial in mk variables. Consider a monomial M of this polynomial; M depends on at most d variables. Since each variable is known to $k - 1$ of the servers (i.e., only one server does not know it), there exists a server that is missing at most $\lfloor d/k \rfloor$ of the variables of M ; we assign the monomial M to this server (if there is more than one server with this property, we pick one arbitrarily).

Suppose, for the moment, that $d = k - 1$. In this case $\lfloor d/k \rfloor = 0$; i.e., the server to which M is assigned knows the assignment $y_{j,h}$ for all the variables $Y_{j,h}$ in M and can actually compute the value of M . The PIR protocol therefore consists of \mathcal{U} picking points \vec{y}_j as in (2), sending each \vec{y}_j to all servers except \mathcal{S}_j , and each server answering \mathcal{U} with the sum (in $\text{GF}(2)$) of all monomials assigned to it. By the

³To be more precise $m = \Theta(dn^{1/d})$. As we treat d and k as constants, we will often ignore multiplicative factors depending on d and k alone.

⁴Alternatively, one can use an encoding with strings of weight *at most* d ; however, for small values of d (e.g., constant) this yields only minor efficiency improvements. Another option is to use the encoding of [17, 8] (denoted **E3** in [8]). This can improve the efficiency of our protocols by a factor of 2^d in some cases, but will further complicate the presentation.

above discussion, the sum of these answers equals x_i . The communication complexity of this protocol is $O(m) = O(n^{1/d}) = O(n^{1/(k-1)})$ bits. More specifically, we have shown:

Claim 3.1 ([20, 8]) *There exists a k -server PIR protocol with query length $O(n^{1/(k-1)})$ and answer length 1.*

Next, consider the case $d = 2k - 1$. Again, assign each monomial M to a server that misses $\lfloor d/k \rfloor = 1$ of the variables of M . Each server \mathcal{S}_j can therefore substitute the values for all but (at most) one of the variables of each monomial M assigned to it. After substituting these values, the sum of the monomials assigned to \mathcal{S}_j can be expressed as a degree-1 polynomial $P_j(Y_{j,1}, \dots, Y_{j,m})$, whose variables $Y_{j,1}, \dots, Y_{j,m}$ are precisely those whose values are unknown to \mathcal{S}_j . Note, however, that if the user could learn all polynomials P_j , then by substituting the correct values $y_{j,h}$ for all their variables and summing up the values of the k polynomials it will get

$$\sum_{j=1}^k P_j(y_{j,1}, \dots, y_{j,m}) = P_x\left(\sum_{j=1}^k \vec{y}_j\right) = P_x(E(i)) = x_i.$$

The PIR protocol starts as before, but this time each server \mathcal{S}_j sends the $m + 1$ coefficients (a single bit each) of the degree-1 polynomial P_j . The communication complexity of this protocol is therefore still $O(m) = O(n^{1/d})$ which, by the choice of d , equals $O(n^{1/(2k-1)})$ bits. To summarize the discussion so far, we have shown how to obtain a PIR protocol with the best known complexity prior to the current work:

Claim 3.2 ([1, 8, 47]) *There exists a k -server PIR protocol with communication complexity $O(n^{1/(2k-1)})$.*

Next, it is useful to note that just further increasing the value of d is of no use. While in such a case each polynomial P_j as above indeed has less variables, it is of a higher degree (i.e., $\lfloor d/k \rfloor$); hence the list of coefficients is no shorter than what we get by choosing $d = 2k - 1$, as above. We emphasize that the amount of information that the user needs about each polynomial P_j is very small (i.e., the value $P_j(\vec{y}_j)$); however, it cannot reveal \vec{y}_j to \mathcal{S}_j as this will expose the value $E(i)$ and hence i .

3.2 The Recursion

The contribution of this paper starts with the following idea to get around the above difficulty. Suppose that we can choose the parameters in a way that each polynomial which the user wants to evaluate is known to several servers. Rather than asking the servers to send the coefficients, we would like the user to *recursively* retrieve the value of this polynomial by using a PIR protocol among the servers sharing the polynomial. Implementing such a recursion is not straightforward, since the general polynomial evaluation problem is not an instance of PIR. To enable efficient recursion, we will need to ensure that the evaluation point held by the user has a *small weight*. Specifically, the evaluation point will be $\vec{z} = E(i)$, whose weight is precisely d (by our definition of E). When this is the case, evaluating the polynomial requires the user to read only a constant number of coefficients from the description of the polynomial, and these can be retrieved via recursive PIR.

Note that in the simple PIR protocol described above, the polynomials P_j are evaluated at points \vec{y}_j which do not generally satisfy the constant-weight property. The main technical difficulty we will need to deal with is that of finding good decompositions of P_x into several polynomials, each known to a strict subset of the servers, such that the user only needs to evaluate these polynomials at the same point \vec{z} .

Towards implementing the strategy described above, we would like to write

$$P_x\left(\sum_{j=1}^k \vec{y}_j\right) = \sum_{V \subseteq [k]} P_V(\vec{z}), \quad (3)$$

where each polynomial P_V is known to every server in the set V , and $\vec{z} = E(i)$ is the assignment known to the user. Before showing how to construct such polynomials P_V , we specify their properties that imply the complexity of our overall solution. We use two parameters λ and k' . The parameter k' is a lower bound on the size of the sets V we will use (except for the sets V of size 1 which will also be used). Each polynomial P_V consists of monomials M in which each of the $|V|$ servers misses at most λ of the variables. Therefore, all but at most $\lambda|V|$ variables of M are known to all servers in V and so after substituting the values known to all servers in V the degree of P_V will be at most $\lambda|V|$. The number of variables on which P_V depends is m (as in P_x). Since the user needs to evaluate P_V at the point $\vec{z} = E(i)$ whose weight is d , it suffices for the user to retrieve the coefficients of the 2^d monomials whose variables are all set to 1 by \vec{z} . (Recall that in our setting of parameters, d is viewed as constant.) Since P_V has $O(m^{\lambda|V|})$ coefficients, the user can retrieve the value $P_x(E(i))$ using 2^d executions of a $|V|$ -server PIR protocol with database of size $O(m^{\lambda|V|}) = O(n^{\lambda|V|/d})$.

Assuming we can indeed find such polynomials P_V with the above properties and assuming that we have a PIR protocol \mathcal{P} with communication complexity $C_{\mathcal{P}}(n, k)$, we get a protocol \mathcal{P}' with communication complexity

$$C_{\mathcal{P}'}(n, k) \leq O_k \left(n^{1/d} + \sum_{\ell=k'}^k \binom{k}{\ell} C_{\mathcal{P}}(n^{\lambda\ell/d}, \ell) \right). \quad (4)$$

(The notation O_k hides multiplicative constants that depend only on k .) An appropriate choice of parameters will ensure, in particular, that $\lambda k/d < 1$ and so \mathcal{P} is applied to shorter strings.

3.3 Constructing the Polynomials P_V for a 3-server PIR Protocol

Before constructing the polynomials P_V for general parameters, we start with the following example. In this example, we show how to construct a 3-server protocol with complexity $O(n^{4/21})$. To this end, we set $k = 3$ and $d = 7$. In this case $\lfloor d/k \rfloor = 2$, therefore for each monomial in P_x there is a server that misses at most 2 variables. This results, without recursion, in a protocol whose complexity is $O(n^{2/7})$ which is worse than the protocol obtained in Claim 3.2. However, by the choice of k and d the following observation holds.

Observation 3.3 *For each monomial M in P_x either (i) there is a server that knows all but at most one of M 's variables, or (ii) there are two servers such that each server knows all but two variables in the monomial M and the third server knows all but three variables in M .*

We will use this observation to write:

$$x_i = P_x\left(\sum_{j=1}^k \vec{y}_j\right) = \sum_{V \subseteq [3], |V|=2} P_V(\vec{z}) + \sum_{j=1}^3 P_j(\vec{y}_j), \quad (5)$$

where each P_V is a polynomial of degree 4 known to the two servers in V and each P_j is a polynomial of degree 1 known to server \mathcal{S}_j . Each polynomial P_V can be evaluated at $\vec{z} = E(i)$ by invoking a 2-server PIR protocol to retrieve its 2^7 relevant coefficients. Using the 2-server protocol of Claim 3.2, this requires $O((m^4)^{1/3}) = O(n^{4/21})$ bits of communication. Since each P_j has degree 1, its entire description can be communicated using $m + 1 = O(n^{1/7})$ bits of communication. Overall, the complexity is $O(n^{4/21})$ as promised.

In the remainder of this example, we show how to construct polynomials P_V and P_j as in Equation (5). It suffices to construct these polynomials for P that consists of a single monomial (and then summing over

all monomials in P_x). Hence, consider, w.l.o.g., $P(Z_1, \dots, Z_m) = Z_1 Z_2 \cdots Z_d$ (instead of P_x). Let

$$Q(\{Y_{j,h}\}_{j \in [3]}) \stackrel{\text{def}}{=} P\left(\sum_{j=1}^3 Y_{j,1}, \dots, \sum_{j=1}^3 Y_{j,7}\right) = \left(\sum_{j=1}^3 Y_{j,1}\right) \left(\sum_{j=1}^3 Y_{j,2}\right) \cdots \left(\sum_{j=1}^3 Y_{j,7}\right).$$

That is, Q is obtained from P by setting $Z_h = \sum_{j=1}^3 Y_{j,h}$. The polynomial Q depends on $kd = 21$ variables and has $k^d = 3^7$ monomials of degree 7 each. Each monomial M is of the form $Y_{j_1,1} \cdots Y_{j_d,d}$.

For every monomial M in Q , consider the set $V(M) \subseteq [3]$ containing all indices of servers that appear at most twice in M . For example, if $M_0 = Y_{3,1} Y_{1,2} Y_{1,3} Y_{2,4} Y_{3,5} Y_{3,6} Y_{2,7}$, then $V(M_0) = \{1, 2\}$. We say that a monomial is *light* if there is an index of a server that appears at most once in M , otherwise we say that it is *heavy*. For instance, the above monomial M_0 is heavy. By Observation 3.3, $|V(M)| = 2$ for each heavy monomial M . The first attempt to define P_V is by assigning all monomials with $V(M) = V$ to V and obtaining P_V by substituting $\{\vec{y}_j\}_{j \notin V}$ in these monomials. For $|V| = 2$, the resulting polynomial P_V has small degree, namely 4, and has few variables, namely $2 \cdot 7$. However, in this case P_V should be evaluated at the point $\{\vec{y}_j\}_{j \in V}$; this point may be of arbitrary weight and, unfortunately, we do not know how to apply the recursion in this case.

To construct a polynomial P_V that can be evaluated at \vec{z} , we group monomials together and replace $\sum_{j_1=1}^3 Y_{j_1,q}$ with Z_q , for example,

$$\sum_{j_1=1}^3 (Y_{j_1,1} Y_{j_2,2} \cdots Y_{j_7,7}) = \left(\sum_{j_1=1}^3 Y_{j_1,1} \right) Y_{j_2,2} \cdots Y_{j_7,7} = Z_1 Y_{j_2,2} \cdots Y_{j_7,7}.$$

To implement the idea of grouping monomials together, denote for a heavy monomial $M = Y_{j_1,1} Y_{j_2,2} \cdots Y_{j_7,7}$

$$\tilde{T}(M) \stackrel{\text{def}}{=} \prod_{j_q \in V(M)} \left(\sum_{j=1}^3 Y_{j,q} \right) \prod_{j_q \notin V(M)} Y_{j_q,q}$$

and

$$T(M) \stackrel{\text{def}}{=} \prod_{j_q \in V(M)} Z_q \prod_{j_q \notin V(M)} y_{j_q,q}.$$

That is, $\tilde{T}(M)$ is a polynomial containing 4^7 monomials, $T(M)$ is a monomial of degree 4 over the variables Z_1, \dots, Z_d (where $\prod_{j_q \notin V(M)} y_{j_q,q}$ is a coefficient), and

$$\tilde{T}(M)(\{\vec{y}_j\}_{j \in [3]}) = T(M)(\vec{z}). \quad (6)$$

Notice that for every heavy monomial M' , the monomial M' is in $\tilde{T}(M)$ if and only if $V(M) = V(M')$ and $\tilde{T}(M) = \tilde{T}(M')$. We say that two heavy monomials M and M' are equivalent if M' is in $\tilde{T}(M)$. This is an equivalence relation amongst the heavy monomials. For each $V \subset [3]$ of size 2 choose one monomial from each equivalence class of the heavy monomials with $V(M) = V$ (there are $\binom{7}{4} = 35$ such equivalence classes for V). Let \mathcal{M}_V be the set of chosen monomials and define

$$P_V(Z_1, \dots, Z_d) \stackrel{\text{def}}{=} \sum_{M \in \mathcal{M}_V} T(M)(\{Z_q\}_{j_q \in V(M)}),$$

and

$$\tilde{P}_V(\{Y_{j,h}\}_{j \in [k], h \in [d]}) \stackrel{\text{def}}{=} \sum_{M \in \mathcal{M}_V} \tilde{T}(M).$$

Finally, define

$$Q' = Q - \sum_{V \subseteq [3], |V|=2} \tilde{P}_V.$$

Every monomial M in Q' is light and each such M is now assigned to a server \mathcal{S}_j that appears at most once in M . The polynomial P_j is the sum of all monomials assigned to \mathcal{S}_j , where the variables $\{Y_{j',q}\}_{j' \in [3], j' \neq j}^{q \in [7]}$ are assigned the values $\{y_{j',q}\}_{j' \in [3], j' \neq j}^{q \in [7]}$. We conclude, using Equation (6), that

$$P\left(\sum_{j=1}^k \vec{y}_j\right) = \sum_{V \subseteq [3], |V|=2} P_V(\vec{z}) + \sum_{j \in [3]} P_{\{j\}}(\vec{y}_j),$$

since every heavy monomial appears in exactly one \tilde{P}_V , and every light monomial either does not appear in Q' or is assigned to exactly one server.

3.4 Constructing the Polynomials P_V for the General Case

Recall that our protocol uses the following parameters: k is the number of servers, k' is a lower bound on the size of the sets V in Equation (3) (which, in turn, is the number of servers used in recursive calls), and λ controls the degree of P_V which is at most $\lambda|V|$ (this degree, in turn, controls the size of the database used in recursive calls). To complete the description of the protocol for general parameters, we provide a specific implementation for the polynomials P_V that, together with the definition of E and P_x , satisfy Equation (3). More precisely, we describe polynomials P_V as above and polynomials P_j of degree 1, such that for every i

$$x_i = P_x(E(i)) = \sum_{V \subseteq [k], |V| \geq k'} P_V(E(i)) + \sum_{j=1}^k P_j(\vec{y}_j). \quad (7)$$

Furthermore, each polynomial P_V can be computed from P_x and $\{\vec{y}_j\}_{j \notin V}$ (this holds for $V = \{j\}$ as well). The construction of the polynomials uses the ideas presented in Section 3.3, however, the construction is more involved.

Consider the polynomial

$$Q_x(\{Y_{j,h}\}_{j \in [k]}^{h \in [m]}) \stackrel{\text{def}}{=} P_x\left(\sum_{j=1}^k Y_{j,1}, \dots, \sum_{j=1}^k Y_{j,m}\right).$$

This is a polynomial with mk variables and degree d . That is, Q_x is obtained from P_x by setting $Z_h = \sum_{j=1}^k Y_{j,h}$. For every monomial M , consider the set $V(M) \subseteq [k]$ containing all servers that appear at most λ times in M .

Before constructing the polynomials P_V , we choose the “correct” value of d . That is, we let d be the maximal value which guarantees that for any monomial M of degree at most d either: (1) there is a server \mathcal{S}_j that knows all but at most one variable in M (in which case this monomial contributes to the corresponding P_j), or (2) the set $V(M)$ has size at least k' .

Claim 3.4 *Let $\lambda, k' \leq k$ be parameters, and $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$. Let M be a monomial of degree at most d in the variables $Y_{j,h}$, where $j \in [k]$ and $h \in [m]$. Then, either there is a server that misses at most one variable in M , or $|V(M)| \geq k'$.*

Proof: Assume that the claim does not hold. That is, every \mathcal{S}_j misses at least two variables in the monomial M (i.e., at least two of the variables $\{Y_{j,h}\}_{h \in [m]}$ appear in M) and at most $k' - 1$ servers miss at most λ variables of M (equivalently at least $k - (k' - 1)$ servers miss at least $\lambda + 1$ variables). Therefore, the number of variables in the monomial is at least $(k - (k' - 1))(\lambda + 1) + (k' - 1) \cdot 2 \geq d + 1$, contradicting the choice of d . \square

Following is the main technical claim underlying our construction.⁵

Claim 3.5 *Let k, λ, k' and d be as in Claim 3.4. Let $P_x(Z_1, \dots, Z_m)$ be a polynomial of degree at most d , and $\vec{z} = \sum_{j=1}^k \vec{y}_j$. Then, there are polynomials $P_V(Z_1, \dots, Z_m)$ for every $V \subseteq [k]$ such that $|V| \geq k'$, and polynomials $P_j(Z_1, \dots, Z_m)$ for $j \in [k]$, such that*

1. *Each polynomial P_V is of degree $\lambda|V|$ and can be computed from P_x and $\{\vec{y}_j\}_{j \notin V}$;*
2. *Each polynomial P_j is of degree 1 and can be computed from P_x and $\{\vec{y}_{j'}\}_{j' \neq j}$;*
3. $P_x(\vec{z}) = \sum_{V \subseteq [k], |V| \geq k'} P_V(\vec{z}) + \sum_{j \in [k]} P_j(\vec{y}_j)$.

Proof: It suffices to prove the claim for polynomials that consist of a single monomial (and then summing over all monomials in P_x). Hence, consider, w.l.o.g., $P(Z_1, \dots, Z_m) = Z_1 Z_2 \cdots Z_d$ (instead of P_x). Let

$$Q(\{Y_{j,h}\}_{j \in [k]}) \stackrel{\text{def}}{=} P\left(\sum_{j=1}^k Y_{j,1}, \dots, \sum_{j=1}^k Y_{j,d}\right) = \left(\sum_{j=1}^k Y_{j,1}\right) \left(\sum_{j=1}^k Y_{j,2}\right) \cdots \left(\sum_{j=1}^k Y_{j,d}\right).$$

The polynomial Q has k^d monomials of degree d each. Each monomial M is of the form $Y_{j_1,1} \cdots Y_{j_d,d}$. Denote

$$\tilde{T}(M) \stackrel{\text{def}}{=} \prod_{j_q \in V(M)} \left(\sum_{j=1}^k Y_{j,q}\right) \prod_{j_q \notin V(M)} Y_{j_q,q}$$

and

$$T(M) \stackrel{\text{def}}{=} \prod_{j_q \in V(M)} Z_q \prod_{j_q \notin V(M)} y_{j_q,q}.$$

The polynomial $\tilde{T}(M)$ is a polynomial obtained by substituting in P each Z_q , where $j_q \in V(M)$, with $\sum_{j=1}^k Y_{j,q}$, while $T(M)$ is a single monomial of degree $\leq \lambda|V(M)|$ whose coefficient $\prod_{j_q \notin V(M)} y_{j_q,q}$ is known to all servers in $V(M)$. Moreover, plugging into both $T(M)$ and $\tilde{T}(M)$ the points $\vec{z}, \vec{y}_1, \dots, \vec{y}_k$ gives the same value.

For every monomial M , denote by $\delta(M)$ the number of variables $Y_{j_q,q}$ in M with $j_q \in V(M)$.

Example 3.6 *Let $k = 3$ and $d = 7$, and consider the monomial $M_0 = Y_{3,1}Y_{1,2}Y_{1,3}Y_{2,4}Y_{3,5}Y_{3,6}Y_{2,7}$. If $\lambda = 2$, then $V(M_0) = \{1, 2\}$ (since each of the indices 1 and 2 appears twice in M_0) and $\delta(M_0) = 4$. However, if $\lambda = 3$, then $V(M_0) = [3]$ and $\delta(M_0) = 7$.*

Below is an algorithm to construct the polynomials P_V as in the claim. The algorithm maintains a polynomial Q' (initially $Q' = Q$) with all the monomials that we need to take care of (monomials may be repeatedly taken out and inserted into Q').

⁵In Section 5 we present an alternative proof of a similar claim based on the inclusion-exclusion principle.

1. Set $Q' = Q$ and for all V set $P_V(Z_1, \dots, Z_m) = 0$.
2. Consider the sets $V(M)$, for all monomials M (currently) in Q' . Pick, among those, a set V such that $V = V(M)$ is of the largest size.
If $|V| < k'$ then STOP.
3. While there is a monomial M such that $V(M) = V$:
 - among these M 's pick M that maximizes $\delta(M)$;
 - set $P_V = P_V + T(M)$ and $Q' = Q' - \tilde{T}(M)$.
4. GOTO 2.

Clearly, the P_V 's are of the desired degree and their sum evaluated at \vec{z} is $P(\vec{z}) - Q'(\vec{z})$ (for Q' that the algorithm halts with). We need to argue that the algorithm halts.

We say that two monomials $M_1 = Y_{j_{1,1}} \cdots Y_{j_{d,d}}$ and $M_2 = Y_{j'_{1,1}} \cdots Y_{j'_{d,d}}$ are equivalent (with respect to $W \subseteq [k]$) if (a) $V(M_1) = V(M_2) = W$; and (b) for each index $q \in [d]$ either j_q, j'_q are both in W or $j_q = j'_q$ (i.e., the same variable $Y_{j_q,q}$ appears in both monomials). Note that this is an equivalence relation and denote $M_1 \equiv M_2$.

Let M_1 be a monomial of $\tilde{T}(M)$ and note the following observations about its structure.

1. $V(M_1) \subseteq V(M)$ (any server not in $V(M) = V$, i.e., one that appears more than λ times in M , appears at least the same number of times in M_1 , by definition of $\tilde{T}(M)$).
2. $\delta(M_1) \leq \delta(M)$ (any variable that does not contribute to $\delta(M)$ does not contribute to $\delta(M_1)$ either).
3. if $V(M_1) = V(M)$ and $\delta(M_1) = \delta(M)$ then, by definition, $M_1 \equiv M$.
4. if $M_2 \equiv M_1$ (with respect to some W) then M_2 must also be in $\tilde{T}(M)$ (by (i), $W \subseteq V(M)$).

It follows that if $M_1 \equiv M_2$ then, at any time during the algorithm, they are either both in Q' or both are not in Q' . This is because it is true at the beginning (all the k^d monomials of the form $Y_{j_{1,1}} \cdots Y_{j_{d,d}}$ are in $Q' = Q$) and whenever Q' is modified by subtracting $\tilde{T}(M)$ for some monomial M then if, say, M_1 is in $\tilde{T}(M)$ then so is M_2 and vice versa.

Using the above observations, we now argue the halting of the algorithm. The idea is that, even though new monomials may be added to Q' when subtracting $\tilde{T}(M)$ in Step 3, such monomials M' either have smaller $V(M')$ or smaller $\delta(M')$ and hence we always make progress. This is because in each application of Step 3 we pick M that maximizes $V(M)$ and among those ones that maximize $\delta(M)$. The monomials added to Q' , when subtracting $\tilde{T}(M)$, satisfy either (a) $V(M') \subset V(M)$ – in which case they will be dealt with in future application of Step 2 if they still exist; or (b) $V(M') = V(M)$ but $\delta(M') < \delta(M)$ – in which case they will be dealt with in future applications of Step 3 if they still exist; or (c) $V(M') = V(M)$ and $\delta(M') = \delta(M)$ – in which case if M is in Q' so is M' and when subtracting $\tilde{T}(M)$ we eliminate both. Once we finish the construction of P_V we do not return to this V anymore.

When the algorithm halts there is no M in Q' for which $|V(M)| \geq k'$. By Claim 3.4 and the choice of d , in such a case for each of these monomials there is at least one \mathcal{S}_j missing at most one variable (from $\{Y_{j,h}\}_{h \in [m]}$). Each such M is now added to a corresponding P_j . Hence, the claim follows. \square

Note that, in spite of the recursion, the resulting protocol can still be implemented as a one-round PIR protocol. The indices of the bits that the user needs in the recursive calls are determined by $E(i)$. Thus, in

the first round, when the user sends its query for i , it can also send its queries for the indices that it needs from every set V . The properties of this protocol are summarized by the next theorem.

Theorem 3.7 *Suppose there is a PIR protocol \mathcal{P} with communication complexity $C_{\mathcal{P}}(n, k)$. Let d, λ, k' be positive integers (which may depend on k) such that $k' < k$ and $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$. Then there is a PIR protocol \mathcal{P}' with communication complexity*

$$C_{\mathcal{P}'}(n, k) = O_k \left(n^{1/d} + \sum_{\ell=k'}^k \binom{k}{\ell} C_{\mathcal{P}}(n^{\lambda\ell/d}, \ell) \right). \quad (8)$$

Remark 3.8 For all PIR protocols from the literature (including the current work)

$$C_{\mathcal{P}}(n^{\lambda\ell/d}, \ell) = O(C_{\mathcal{P}}(n^{\lambda k'/d}, k'))$$

for $\ell \geq k'$. Thus, the sum in Equation (8) is dominated by its first term.

Example 3.9 *We demonstrate how to get the first two improved protocols from Figure 1. In the 3-server case, we set $\lambda = k' = 2$ and $d = 7$ (as detailed in Section 3.3). By using a 2-server protocol with complexity $O(n^{1/3})$ (see Claim 3.2) the communication complexity is $O(n^{1/7} + (n^{4/7})^{1/3}) = O(n^{4/21})$. In the 4-server case, we can rely on the above protocol and use $\lambda = 2, k' = 3$, and $d = 9$ to obtain communication complexity of $O(n^{1/9} + (n^{6/9})^{4/21}) = O(n^{8/63})$.*

3.5 Analyzing the Communication Complexity of the Protocol

The above discussion yields a recursive complexity analysis. Below we get a specific bound by choosing appropriate parameters. This analysis is somewhat crude and is mainly intended for large values of k (which are still viewed as constants). For small values of k , one should be more careful. The results specified in Figure 1 are derived by choosing optimal values for the parameters.

Lemma 3.10 *For every positive integer i , there is a PIR protocol \mathcal{P}_i such that $C_{\mathcal{P}_i}(n, k) = O_k(n^{2/(ik)})$ for every constant $k \geq (i - 1)!$.*

Proof: The proof is by induction on i . The first nontrivial case is $i = 3$, in which the lemma follows from Claim 3.2. For the induction step, suppose that $i \geq 3$ and there exists a PIR protocol \mathcal{P}_i such that $C_{\mathcal{P}_i}(n, k) = O_k(n^{2/(ik)})$ for every $k \geq (i - 1)!$. Using Theorem 3.7, we construct \mathcal{P}_{i+1} such that $C_{\mathcal{P}_{i+1}}(n, k) = O_k(n^{2/((i+1)k)})$ for every $k \geq i!$. Let

$$\begin{aligned} k' &= \lfloor k/i \rfloor \geq (i - 1)!, \\ \lambda &= \lceil i/2 \rceil \end{aligned}$$

(in particular, $\lambda \geq 2$), and

$$d = (\lambda + 1)k - (\lambda - 1)k' \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2),$$

as required for applying Theorem 3.7. Note that $C_{\mathcal{P}_i}(n^{\lambda\ell/d}, \ell) = O_k(n^{2\lambda/(id)})$ for every $\ell \geq k'$. Thus, $C_{\mathcal{P}_{i+1}}(n, k) = O_k(n^{1/d} + n^{2\lambda/(id)}) = O_k(n^{2\lambda/(id)})$, where the last equality follows from the fact that $2\lambda/i \geq 1$. To complete the proof, it suffices to prove that $2\lambda/(id) \leq 2/((i + 1)k)$. Indeed,

$$\frac{\lambda}{id} \leq \frac{\lambda}{i((\lambda + 1)k - (\lambda - 1)k/i)} \leq \frac{\lambda}{k(\lambda i + (i - \lambda + 1))} \leq \frac{\lambda}{k(\lambda i + \lambda)} = \frac{1}{k(i + 1)},$$

where the first inequality is by the choice of d and k' and the last inequality is by the choice of λ . \square

Corollary 3.11 *There exists a PIR protocol \mathcal{P} such that $C_{\mathcal{P}}(n, k) = O_k(n^{2 \log \log k / (k \log k)})$, for every $k \geq 3$.*

Proof: For $k \geq 3$ the protocol \mathcal{P} executes the protocol \mathcal{P}_i guaranteed by Lemma 3.10, where $i = \lceil \log k / \log \log k \rceil$. Then, $(i - 1)! \leq (i - 1)^{i-1} \leq (\log k)^{\log k / \log \log k} = k$, and $C_{\mathcal{P}}(n, k) = C_{\mathcal{P}_i}(n, k) = O_k(n^{2 \log \log k / (k \log k)})$. \square

In Appendix B we show that the above analysis is essentially optimal. This is not to say that there are no other protocols that can do better; it only states that within the freedom that our protocol has in choosing the parameters λ and k' , the above choice, that achieves complexity of $n^{O(\log \log k / (k \log k))}$, is essentially the best.

4 PIR Protocols with Short Answers and Locally Decodable Codes

In this section, we obtain efficient PIR protocols in which the answer of each server consists of a single bit; we refer to such protocols as “binary protocols.” We start by noting that the protocols from the previous section can be transformed in a *generic* way to binary protocols with related complexity. Specifically, given any *linear* k -server PIR protocol in which the total communication with each server is $c(n)$, it is possible to construct a $2k$ -server binary protocol with query length $c(n)$ [27].⁶

Thus, there is a binary k -server PIR protocol with query length $n^{O(\log \log k / (k \log k))}$. Below is a direct construction which improves the constants in the above exponent. In particular, while the generic transformation improves over the best previously known *binary* protocols only for $k \geq 6$, the following direct construction gives the first improvement when $k = 4$.

Theorem 4.1 *Suppose there is a PIR protocol \mathcal{P} with query length $Q_{\mathcal{P}}(n, k)$ and answer length $A_{\mathcal{P}}(n, k)$. Let d, λ, k' be positive integers (which may depend on k) such that $k' < k$ and $d \leq (\lambda + 1)k - \lambda k' + (\lambda - 1)$. Then, there is a PIR protocol \mathcal{P}' with query length*

$$Q_{\mathcal{P}'}(n, k) = O_k \left(n^{1/d} + \sum_{\ell=k'}^k \binom{k}{\ell} Q_{\mathcal{P}}(n^{\lambda \ell / d}, \ell) \right) \quad (9)$$

and answer length $O_k(\sum_{\ell=k'}^k \binom{k}{\ell} A_{\mathcal{P}}(n^{\lambda \ell / d}, \ell))$. Furthermore, if \mathcal{P} is binary and linear then there is a binary linear \mathcal{P}' with query length as above.

Proof: As in the proof of Claim 3.4, the condition $d \leq (\lambda + 1)k - \lambda k' + \lambda - 1$ guarantees that in every monomial M , either $V(M) \geq k'$ or there is a server that knows *all* variables in the monomial. Thus, following the proof of Claim 3.5, the current (more strict) setting of the parameters allows to obtain a variant of Claim 3.5 where the polynomials P_j are of degree 0. It therefore suffices for each server to send one bit to the user in addition to the answers in the recursive calls. If \mathcal{P} is a binary, linear PIR protocol then the user

⁶It is easy to verify that the protocols constructed in the previous section are in fact linear. The transformation to a binary protocol may proceed as follows. The user generates queries q_1, \dots, q_k as in the original protocol and sends each q_j to both \mathcal{S}_j and \mathcal{S}_{k+j} ; each of them generates the corresponding answer a_j but does not send it back. Instead, the user privately retrieves the exclusive-or of the bits that it needs from a_j using the following procedure. The user needs to learn the inner product (in $\text{GF}(2)$) of a_j with some vector b_j it knows. To this end it sends a random vector r_j of length $|a_j|$ to \mathcal{S}_j and $r_{j+k} = r_j - b_j$ to \mathcal{S}_{j+k} . Each of the two servers replies with the inner product of a_j and the received vector, allowing the user to recover $\langle b_j, a_j \rangle$ by adding (in $\text{GF}(2)$) the two received bits. In the language of codes, this may be viewed as a *concatenation* of a locally-decodable code over a large alphabet with the Hadamard code.

recovers x_i in \mathcal{P} by computing the exclusive-or of the bits that it gets (one from each server). Thus, in the protocol \mathcal{P}' it suffices for each server j to send the exclusive-or of all the bits it would send in the calls to \mathcal{P} together with the bit describing the (degree 0) polynomial P_j . \square

Note that in Theorem 4.1 we require that $d \leq (\lambda + 1)k - \lambda k' + (\lambda - 1)$, while in Theorem 3.7 we require that $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$. Thus, the communication complexity in the protocol of Theorem 4.1 is worse than the communication complexity in the protocol of Theorem 3.7. However, the answer complexity in the protocol of Theorem 4.1 is $O_k(\sum_{\ell=k'}^k \binom{k}{\ell} A_{\mathcal{P}}(n^{\lambda \ell/d}, \ell))$. This is better than the answer complexity in the protocol of Theorem 3.7 which is $O_k(n^{1/d} + \sum_{\ell=k'}^k \binom{k}{\ell} A_{\mathcal{P}}(n^{\lambda \ell/d}, \ell))$.

Example 4.2 We illustrate the use of Theorem 4.1 for obtaining the first two improvements over previous protocols. As a basis we can use the case $k = 3$, for which the best known binary protocol has query length $O(n^{1/2})$ (see Claim 3.1). For $k = 4$ we let $\lambda = 1, k' = 3$, and $d = 5$, and get a 4-server binary protocol with query length $O(n^{1/5} + (n^{3/5})^{1/2}) = O(n^{3/10})$. In the 5-server case we let $\lambda = 1, k' = 4, d = 6$ and get a binary protocol with query length $O(n^{1/5})$.

4.1 Application to locally decodable codes

A binary code $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ is said to be (k, δ, ρ) -locally decodable if every bit x_i of x can be decoded from $y = C(x)$ with success probability $\geq 1/2 + \rho$ by reading k (randomly chosen) bits of y , even if up to a δ -fraction of y was adversarially corrupted. A k -query binary locally-decodable code is a family of $(k, \delta(n), \rho(n))$ -locally decodable codes $C_n : \{0, 1\}^n \rightarrow \{0, 1\}^{m(n)}$ such that $\delta(n), \rho(n)$ are lower bounded by some positive constant, independent of n .

Given a binary k -server PIR protocol with query length $c(n)$, it is possible to construct a k -query binary locally-decodable code of length $O(k \cdot 2^{c(n)})$ [33]. If the query to each server is uniformly distributed over its domain, as is the case for the protocols we obtain, the encoding of a string $x \in \{0, 1\}^n$ can be defined as the concatenation of the servers' answers to all possible queries, i.e., $\mathcal{A}(k, j, x, q)$ for all $j \in [k]$ and all queries q . Thus, we have:

Corollary 4.3 *There is a constant c such that for every k there is a k -query binary locally-decodable code of length $O(2^{n^{c \log \log k / (k \log k)}})$.*

5 An Abstract Framework

In this section we describe an abstract framework which generalizes the specific protocol from Section 3 and captures the scope of the underlying technique. Using this framework, we provide an alternative proof of Theorem 3.7 – our recursive PIR construction. We start by formulating a general linear algebra problem that lies in the core of Claim 3.5.

Fix some field \mathbb{F} (where $\mathbb{F} = \text{GF}(2)$ by default).⁷ We consider the linear space of polynomials over \mathbb{F} in the dk variables $Y_{j,h}$, where $j \in [k], h \in [d]$.⁸ In fact, we will only be interested in the subspace spanned by the k^d monomials of the form $Y_{j_1,1} Y_{j_2,2} \cdots Y_{j_d,d}$, for $j_1, \dots, j_d \in [k]$.

We use Z_h as an abbreviation for the sum $\sum_{j=1}^k Y_{j,h}$. Following the terminology from [8], a *block* is a polynomial which can be expressed as a product of sums Z_h and variables $Y_{j,h}$ in which each index $h \in [d]$

⁷All of the concrete results in this section hold over an arbitrary field \mathbb{F} and in particular over $\mathbb{F} = \text{GF}(2)$. However, since our framework may have different power over different fields we prefer a more general treatment.

⁸Notice that h here is an index whose value is at most d , unlike previous sections where $h \in [m]$.

appears exactly once. For example, every $\tilde{T}(M)$ from the proof of Claim 3.5 is a block. Note that if b is a block, then its representation as such a product must be unique. We will identify this representation of b with a product \hat{b} of d variables $Y_{j,h}$ or Z_h . One can view the set of monomials (in the variables $Y_{j,h}$) involved in a block as a sub-cube of $[k]^d$ obtained by restricting a subset of the d coordinates. Using this view, we denote a block by a d -tuple over $[k] \cup \{*\}$. That is, if $Y_{j,h}$ appears in the block, then the h th coordinate is j and if Z_h appears in the block, then the h th coordinate is $*$.

Example 5.1 For $k = 2$ and $d = 3$ we identify between the block $b = (Y_{1,1} + Y_{2,1}) \cdot Y_{1,2} \cdot (Y_{1,3} + Y_{2,3})$, its representation $\hat{b} = Z_1 Y_{1,2} Z_3$, and the 3-tuple $(*, 1, *)$.

Next we define two important parameters of a block that will determine the complexity of a PIR protocol using it.

Definition 5.2 ($\delta(b)$ and $V(b)$) For each block b , let $\delta(b)$ denote the number of $*$ symbols in the corresponding d -tuple (alternatively, the “dimension” of the corresponding sub-cube, or the number of Z_h variables occurring in \hat{b}). We let $V(b)$ denote the set of indices $j \in [k]$ which do not occur in the corresponding d -tuple (alternatively, in variables $Y_{j,h}$ appearing in \hat{b}).

Example 5.3 If $d = 5, k = 5$, and $b = (*, 5, 5, *, 2) = Z_1 Y_{5,2} Y_{5,3} Z_4 Y_{2,5}$ then $\delta(b) = 2$, and $V(b) = \{1, 3, 4\}$.

Jumping ahead, in the context of the PIR application the block b will be used by the servers in $V(b)$ to construct a polynomial of degree $\delta(b)$ in the variables Z_1, \dots, Z_m .

We now define two key properties of sets of blocks, generalizing the corresponding notions from [8].

Definition 5.4 (Spanning and covering sets of blocks) Let \mathcal{B} be a set of blocks with parameters d, k (that is, $\mathcal{B} \subseteq ([k] \cup \{*\})^d$). We say that \mathcal{B} is spanning if the blocks in \mathcal{B} , viewed as polynomials (over \mathbb{F}) in the dk variables $Y_{j,h}$, span the block $(*, *, \dots, *) = Z_1 Z_2 \cdots Z_d = \sum_{\vec{w} \in [k]^d} Y_{w_1,1} Y_{w_2,2} \cdots Y_{w_d,d}$. We say that \mathcal{B} is covering if each of the k^d monomials occurs in some block $b \in \mathcal{B}$.

Clearly, any spanning block set must also be covering. The following example shows that the converse does not hold.

Example 5.5 Let $d = k = 2$. Then $\mathcal{B} = \{b_1 = (1, 1), b_2 = (*, 2), b_3 = (2, 1)\}$ is spanning, since

$$(*, *) = (Y_{1,1} + Y_{2,1})(Y_{1,2} + Y_{2,2}) = Y_{1,1}Y_{1,2} + (Y_{1,1} + Y_{2,1})Y_{2,2} + Y_{2,1}Y_{1,2} = b_1 + b_2 + b_3.$$

On the other hand, it is not hard to verify that $\mathcal{B}' = \{(1, *), (*, 2), (2, 1)\}$ is not spanning, even though it is covering (as each of the four monomials is involved in some block in \mathcal{B}').

The following set of blocks was used in [8] to construct a PIR protocol with communication complexity $O(k^3 n^{1/(2k-1)})$.

Example 5.6 If $d = 2k - 1$, then the set \mathcal{B} of all blocks b such that $\delta(b) \leq 1$ and $|V(b)| \geq 1$ (that is, the set of blocks with at most one $*$ that miss at least one index) is spanning. We first show that \mathcal{B} is covering. This holds because any tuple in $[k]^d$ (representing a monomial) contains some index j that occurs at most once (since $d < 2k$). Replacing this index by a $*$, we get a block in \mathcal{B} covering the monomial.

To show that \mathcal{B} is also spanning, it suffices to show that every monomial in $[k]^d$ is spanned by blocks in \mathcal{B} . Indeed, in a monomial b that is not in \mathcal{B} each index occurs at least once. It follows that there is an index

j that occurs exactly once in b . Define the block b' as the block b where j is replaced by $*$ and the block $b_{j'}$, for $j' \in [k] \setminus \{j\}$, as the block b where j is replaced by j' . Note that both b' and $b_{j'}$ are in \mathcal{B} , as j does not occur in all these blocks, and moreover $b = b' - \sum_{j' \in [k] \setminus \{j\}} b_{j'}$. Thus, b is spanned by blocks in \mathcal{B} . For instance, if $k = 3$ and $d = 5$ then $(1, 1, 2, 2, 3)$ can be written as $(1, 1, 2, 2, 3) = (1, 1, 2, 2, *) - (1, 1, 2, 2, 1) - (1, 1, 2, 2, 2)$.

Generalizing Theorem 3.7, it is possible to use any spanning block set \mathcal{B} for reducing k -server PIR to instances of PIR with a smaller number of servers. This is formalized by the following theorem.

Theorem 5.7 *Suppose there is a PIR protocol \mathcal{P} with communication complexity $C_{\mathcal{P}}(n, k)$. Then, for any $d = d(k)$ and spanning block set $\mathcal{B} = \mathcal{B}(d, k)$ with parameters d, k there is a PIR protocol \mathcal{P}' with communication complexity*

$$C_{\mathcal{P}'}(n, k) = O_k \left(n^{1/d} + \sum_{b \in \mathcal{B}} C_{\mathcal{P}}(n^{\delta(b)/d}, |V(b)|) \right). \quad (10)$$

Proof: Fix k, d , and a spanning block set \mathcal{B} . Suppose for simplicity that $\sum_{b \in \mathcal{B}} b = (*, *, \dots, *)$. (In the default case where $\mathbb{F} = \text{GF}(2)$ this can be assumed without loss of generality.) The protocol \mathcal{P}' proceeds similarly to the protocol described in Section 3. We include the full details for self-containment.

We associate with each $i \in [n]$ a distinct weight- d vector $E(i) \in \{0, 1\}^m$, where $m = \Theta(n^{1/d})$, and represent a database x by an m -variate polynomial

$$P_x(Z_1, \dots, Z_m) \stackrel{\text{def}}{=} \sum_{j=1}^n x_j \prod_{\ell: E(j)_\ell=1} Z_\ell,$$

where $E(j)_\ell$ is the ℓ th bit of $E(j)$. Using this representation, we have $P_x(E(i)) = x_i$ for every x and i . The user transforms its selection index i into the vector $\vec{z} = E(i)$ and splits \vec{z} into k additive shares \vec{y}_j . That is, \vec{y}_j are random subject to the restriction that $\sum_{j=1}^k \vec{y}_j = \vec{z}$. The user sends to each server h the $k-1$ vectors \vec{y}_j such that $j \neq h$.

The goal is to reveal $P_x(\vec{z}) = P_x(\vec{y}_1 + \dots + \vec{y}_k)$ to the user while hiding \vec{z} from any individual server. To this end, we will use the spanning block set \mathcal{B} to write

$$P_x(\vec{y}_1 + \dots + \vec{y}_k) = \sum_{b \in \mathcal{B}} P_b(\vec{z}), \quad (11)$$

where each P_b is a polynomial of degree at most $\delta(b)$ known to all servers in $V(b)$. (Each polynomial P_b will be determined by servers in $V(b)$, as specified below, based on the database x and the points \vec{y}_j such that $j \notin V(b)$.) Since \vec{z} is of weight d , the user only needs to learn a constant number (at most 2^d) of coefficients from each polynomial P_b in order to evaluate it at \vec{z} . Moreover, the description of each P_b includes at most $m^{\delta(b)} = O(n^{\delta(b)/d})$ coefficients. Thus, each $P_b(\vec{z})$ can be retrieved using a constant number of invocations of $\mathcal{P}(O(n^{\delta(b)/d}), |V(b)|)$. Together with the $O(m) = O(n^{1/d})$ bits required for communicating the points \vec{y}_j to the servers, we get the communication complexity specified in Equation (10).

It remains to show how to obtain the decomposition of Equation (11). For this, we use the following notation. Denote by $\vec{y}|_{E(j)}$ the d -tuple obtained by restricting a vector $\vec{y} \in \{0, 1\}^m$ to the entries h such that $E(j)_h = 1$. We view a block b as a degree- d polynomial

$$b((Y_{1,1}, \dots, Y_{1,d}), (Y_{2,1}, \dots, Y_{2,d}), \dots, (Y_{k,1}, \dots, Y_{k,d}))$$

over dk variables, and its representation \hat{b} as a degree- d monomial $\hat{b}(\vec{Y}; \vec{Z})$ over the $dk + d$ variables $\vec{Y} = (Y_{j,h})$ and $\vec{Z} = (Z_1, \dots, Z_m)$. (Recall that \hat{b} is the representation of b as a product of $\delta(b)$ variables Z_h and $d - \delta(b)$ variables $Y_{j,h}$.) Finally, let $M(Z_1, \dots, Z_d) \stackrel{\text{def}}{=} Z_1 Z_2 \cdots Z_d$.

Using the above notation, we have:

$$\begin{aligned}
P_x(\vec{y}_1 + \dots + \vec{y}_k) &= \sum_{j=1}^n x_j \cdot M(\vec{y}_1|_{E(j)} + \vec{y}_2|_{E(j)} + \dots + \vec{y}_k|_{E(j)}) \\
&= \sum_{j=1}^n x_j \cdot \sum_{b \in \mathcal{B}} b(\vec{y}_1|_{E(j)}, \vec{y}_2|_{E(j)}, \dots, \vec{y}_k|_{E(j)}) \\
&= \sum_{b \in \mathcal{B}} \sum_{j=1}^n x_j \cdot b(\vec{y}_1|_{E(j)}, \vec{y}_2|_{E(j)}, \dots, \vec{y}_k|_{E(j)}) \\
&= \sum_{b \in \mathcal{B}} \sum_{j=1}^n x_j \cdot \hat{b}(\vec{y}_1|_{E(j)}, \vec{y}_2|_{E(j)}, \dots, \vec{y}_k|_{E(j)}; \vec{Z}|_{E(j)}).
\end{aligned}$$

It follows that Equation (11) is satisfied by letting

$$P_b(\vec{Z}) = \sum_{j=1}^n x_j \cdot \hat{b}(\vec{y}_1|_{E(j)}, \vec{y}_2|_{E(j)}, \dots, \vec{y}_k|_{E(j)}; \vec{Z}|_{E(j)}).$$

The degree of P_b is at most $\delta(b)$ since \hat{b} depends on $\delta(b)$ variables Z_h . Finally, we need to show that P_b is known to all servers in $V(b)$. This follows from the fact that (by definition of $V(b)$) if \hat{b} depends on a variable $Y_{j,h}$ then all servers in $V(b)$ hold the vector \vec{y}_j . \square

We note that Theorem 5.7 gives, in a sense, a closed-form expression for the best communication complexity attainable by the current approach. The main difficulty, however, is in finding appropriate choices for the spanning block set \mathcal{B} that optimize the overall complexity. We do not know if, using Theorem 5.7, it is possible to construct a protocol whose complexity is better than the concrete protocol presented in Theorem 3.7 and Corollary 3.11. In Appendix B.2, we show that using Theorem 5.7 one *cannot* obtain a protocol whose complexity is better than $n^{O(1/(k \log k))}$. Thus, the generalization offered by the abstract framework only leaves a modest room for improvement.

In the rest of the section we obtain an alternative derivation of Theorem 3.7 using Theorem 5.7 and an appropriate spanning set of blocks. We naturally identify a block b with the corresponding set of $k^{\delta(b)}$ monomials (over the variables $Y_{j,h}$). Accordingly, the *intersection* $b_1 \cap b_2$ of two blocks is the polynomial which includes all monomials appearing in both blocks. Note that any nonempty intersection of blocks is a block. For instance, $(1, *, *) \cap (*, *, 2) = (1, *, 2)$. Recall that a block set \mathcal{B} is *covering* if each of the k^d monomials occurs in some block $b \in \mathcal{B}$. We have seen in Example 5.5 that covering does not imply spanning. However, it follows from the inclusion-exclusion principle that if \mathcal{B} is covering and is also *closed under intersections* then it is in fact spanning. We prove this claim in a slightly more general form, which will be useful for constructing good spanning sets of blocks.

Lemma 5.8 *Suppose that a block set \mathcal{B} satisfies the following properties:*

1. \mathcal{B} is covering.
2. For any $b_1, b_2 \in \mathcal{B}$, the intersection $b_1 \cap b_2$ is spanned by blocks in \mathcal{B} .

Then \mathcal{B} is spanning.

Proof: Consider the polynomial

$$\sum_{b_i \in \mathcal{B}} b_i - \sum_{b_i, b_j \in \mathcal{B}} (b_i \cap b_j) + \sum_{b_i, b_j, b_k \in \mathcal{B}} (b_i \cap b_j \cap b_k) - \sum_{b_i, b_j, b_k, b_\ell \in \mathcal{B}} (b_i \cap b_j \cap b_k \cap b_\ell) + \dots$$

where the m -th summation is over all sets of *distinct* m blocks from \mathcal{B} . By the inclusion-exclusion principle, the above polynomial is equal to the sum of all monomials that appear in *some* block of \mathcal{B} .⁹ Since \mathcal{B} is covering, this sum is equal to the sum of all k^d monomials. It remains to show that the above polynomial is spanned by \mathcal{B} (as it contains intersections of more than two blocks). Indeed, it follows inductively from property (2) and the identity $b_1 \cap (b_2 + b_3) = (b_1 \cap b_2) + (b_1 \cap b_3)$ that the intersection of an arbitrary number of blocks from \mathcal{B} is spanned by \mathcal{B} . \square

We now demonstrate the use of Lemma 5.8 for obtaining our improved 3-server protocol.

Example 5.9 Let $k = 3$ and $d = 7$ as in Section 3.3. Let \mathcal{B} be the set of blocks b such that $\delta(b) \leq 4$ and every server index $j \in [k] \setminus V(b)$ occurs in b at least 3 times. That is, each block $b \in \mathcal{B}$ either contains one index occurring at least 3 times (in which case $|V(b)| = 2$ and $\delta(b) \leq 4$) or contains two indices each occurring at least 3 times (in which case $|V(b)| = 1$ and $\delta(b) \leq 1$). For instance, $(3, 3, 3, *, *, *, *) = Y_{3,1}Y_{3,2}Y_{3,3}Z_4Z_5Z_6Z_7$ and $(3, 3, 3, *, 1, 1, 1)$ are blocks in \mathcal{B} , whereas $(3, 3, 3, *, *, 1, 1)$ is not (since $j = 1$ occurs only twice). First, we argue that \mathcal{B} is covering. This follows from the fact that in any vector from $[3]^7$ representing a monomial, there must be some index $j \in [3]$ occurring at least 3 times. Thus, replacing all other indices by ‘*’ we get a representation of a block in \mathcal{B} covering the monomial.

Next, we argue that if $b_1, b_2 \in \mathcal{B}$ have a nonempty intersection $b = b_1 \cap b_2$, then $b \in \mathcal{B}$. Indeed, every index occurs in b in some coordinate if and only if it occurs in b_1 or in b_2 in the same coordinate. Thus, every index appears in b at least 3 times and $\delta(b) \leq \min\{\delta(b_1), \delta(b_2)\} \leq 4$. For instance, $(1, 1, 1, *, *, *, *) \cap (*, *, 1, 1, 1, *, *) = (1, 1, 1, 1, 1, *, *)$, and $(1, 1, 1, *, *, *, *) \cap (*, *, *, *, 2, 2, 2) = (1, 1, 1, *, 2, 2, 2)$, while $(1, 1, *, *, *, *, 1) \cap (*, *, *, *, 2, 2, 2)$ is empty. Thus, by Lemma 5.8 the set \mathcal{B} is spanning.

Finally, using Theorem 5.7 and a 2-server protocol \mathcal{P} with complexity $O(n^{1/3})$ for $k = 2$ servers (and a trivial complexity of n for $k = 1$ servers), we get a 3-server protocol \mathcal{P}' with complexity $O(n^{4/21})$. Indeed,

$$\begin{aligned} C_{\mathcal{P}'}(n, 3) &= O\left(n^{1/7} + \sum_{b \in \mathcal{B}} C_{\mathcal{P}}(n^{\delta(b)/7}, |V(b)|)\right) \\ &= O\left(n^{1/7} + C_{\mathcal{P}}(n^{4/7}, 2) + C_{\mathcal{P}}(n^{1/7}, 1)\right) \\ &= O(n^{1/7} + (n^{4/7})^{1/3}) = O(n^{4/21}), \end{aligned}$$

where the second equality follows from the fact that the most “expensive” blocks in \mathcal{B} are ones with $\delta(b) = 4$ and $|V(b)| = 2$.

We conclude by presenting our general spanning set of blocks, corresponding to Claim 3.5.

Claim 5.10 Let k, λ, k', d be such that $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$ (as in Claim 3.4). Then, the following set \mathcal{B} is spanning. The block set \mathcal{B} includes blocks of two types:

⁹This equality holds regardless of the underlying field. In our default case of polynomials over $\text{GF}(2)$, there is no significance to the alternation of signs.

Type 1 blocks. Every block b such that $\delta(b) \leq 1$ and $|V(b)| \geq 1$.

Type 2 blocks. Every block b such that

1. each $j \in [k] \setminus V(b)$ occurs in b more than λ times,
2. $\delta(b) \leq \lambda|V(b)|$, and
3. $|V(b)| \geq k'$.

Proof: We again rely on Lemma 5.8. The covering condition follows from the same counting argument as in Claim 3.4. Indeed, for any monomial in $[k]^d$ either there is a server index j occurring at most once (in which case it is covered by the block of type 1 obtained by replacing the possible occurrence of j by ‘*’) or alternatively there are at least k' indices j each occurring at most λ times (in which case it is covered by the block of type 2 obtained by replacing all such indices by ‘*’).

For the intersection condition we consider the following cases. An intersection involving a block of type 1 is spanned by blocks of type 1, since this intersection is either itself of type 1, or alternatively it is a monomial which contains an index that occurs once. In the latter case it is spanned by k blocks of type 1 (similarly to Example 5.6). For the case of two blocks b_1, b_2 of type 2, their intersection $b = b_1 \cap b_2$ clearly satisfies conditions (1) and (2) of blocks of type 2. Now, if b also satisfies condition (3) then we are done. We argue that if b does not satisfy condition (3), then it is spanned by blocks of type 1. Indeed, in the latter case we have less than k' server indices j missing from b (since (3) is not satisfied) and each index occurring in b occurs there more than λ times (since (1) is satisfied). Thus, each monomial in b has less than k' indices j occurring at most λ times. By the choice of parameters, each such monomial has an index occurring at most once (see previous paragraph), which means (again) that it is also spanned by blocks of type 1. \square

Theorem 3.7 now follows by combining Theorem 5.7 with the spanning set of blocks given by Claim 5.10.

Acknowledgments. We thank Omer Barkol, Don Coppersmith, Enav Weinreb, David Woodruff, and Sergey Yekhanin for helpful discussions and 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-Verlag, 1997.
- [2] L. Babai, A. Gál, P. G. Kimmel, and S. V. Lokam. Communication complexity of simultaneous messages. *SIAM J. on Computing*, 33(1):137 – 166, 2003.
- [3] L. Babai, P. G. Kimmel, and S. V. Lokam. Simultaneous messages vs. communication. In E. W. Mayr and C. Puech, editors, *12th Annual Symposium on Theoretical Aspects of Computer Science*, volume 900 of *Lecture Notes in Computer Science*, pages 361–372. Springer-Verlag, 1995. Journal version in [2].
- [4] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In C. Choffrut and T. Lengauer, editors, *STACS '90, 7th 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] R. Beigel, L. Fortnow, and W. Gasarch. Nearly tight bounds for private information retrieval systems. Technical Report TR03-087, Electronic Colloquium on Computational Complexity, <http://www.eccc.uni-trier.de/eccc/>, 2003.
- [7] A. Beimel and Y. Ishai. Information-theoretic private information retrieval: A unified construction. In P. G. Spirakis and J. van Leeuwen, 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.
- [8] A. Beimel, Y. Ishai, and E. Kushilevitz. General constructions for information-theoretic private information retrieval. *J. of Computer and System Sciences*, 71(2):213–247, 2005. This is a full version of [30] and [7].
- [9] A. Beimel, Y. Ishai, E. Kushilevitz, and J. F. Raymond. Breaking the $O(n^{\frac{1}{2k-1}})$ barrier for information-theoretic private information retrieval. In *Proc. of the 43rd 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. *J. of Cryptology*, 17(2):125–151, 2004. Preliminary version: M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 56–74. Springer, 2000.
- [11] A. Beimel and Y. Stahl. Robust information-theoretic private information retrieval. In S. Cimato, C. Galdi, and G. Persiano, editors, *3rd Conf. on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 326–341. Springer-Verlag, 2002.
- [12] 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-Verlag, 1999.
- [13] R. Canetti, Y. Ishai, R. Kumar, M. K. Reiter, R. Rubinfeld, and R. N. Wright. Selective private function evaluation with applications to private statistics. In *Proc. of the 20th ACM symp. on Principles of Distributed Computing*, pages 293 – 304, 2001.
- [14] Y.-C. Chang. Single database private information retrieval with logarithmic communication. In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004*, volume 3108 of *Lecture Notes in Computer Science*, pages 50 – 61. Springer-Verlag, 2004.
- [15] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of the 29th ACM Symp. on the Theory of Computing*, pages 304–313, 1997.
- [16] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. Technical report, Department of Computer Science, Technion, 1997.

- [17] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of the 36th IEEE Symp. on Foundations of Computer Science*, pages 41–51, 1995. Journal version: *J. of the ACM*, 45:965–981, 1998.
- [18] R. Cramer, I. Damgård, and Y. Ishai. Share conversion, pseudorandom secret-sharing and applications to secure distributed computing. In J. Kilian, editor, *Proc. of the Second Theory of Cryptography Conference – TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362. Springer-Verlag, 2005.
- [19] A. Deshpande, R. Jain, T. Kavita, V. Lokam, and J. Radhakrishnan. Better lower bounds for locally decodable codes. In *Proc. of the 17th IEEE Conf. on Computational Complexity*, pages 184–193, 2002.
- [20] 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 ACM Symp. on Principles of Distributed Computing*, pages 91–100, 1998.
- [21] Z. Dvir and A. Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. In *Proc. of the 37th ACM Symp. on the Theory of Computing*, pages 592–601, 2005.
- [22] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright. Secure multiparty computation of approximations. In P. G. Spirakis and J. van Leeuwen, editors, *Proc. of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 927–938. Springer-Verlag, 2001.
- [23] W. Gasarch. A survey on private information retrieval. *Bulletin of the European Association for Theoretical Computer Science*, 82:72–107, 2004. Also can be found at: <http://www.cs.umd.edu/~gasarch/pir/pir.html>.
- [24] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *Proc. of the 32nd International Colloquium on Automata, Languages and Programming*, pages 803–815, 2005.
- [25] 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-Verlag, 1998.
- [26] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. of Computer and System Sciences*, 60(3):592–629, 2000.
- [27] O. Goldreich. Personal communication, 2000.
- [28] O. Goldreich. Short locally testable codes and proofs (survey). Technical Report TR05-014, Electronic Colloquium on Computational Complexity, <http://www.eccc.uni-trier.de/eccc/>, 2005. <http://www.eccc.uni-trier.de/eccc/>.
- [29] O. Goldreich, H. Karloff, L. J. Schulman, and L. Trevisan. Lower bounds for linear locally decodable codes and PIR. In *Proc. of the 17th IEEE Conf. on Computational Complexity*, pages 175–183, 2002.
- [30] Y. Ishai and E. Kushilevitz. Improved upper bounds on information theoretic private information retrieval. In *Proc. of the 31st ACM Symp. on the Theory of Computing*, pages 79 – 88, 1999.

- [31] T. Itoh. Efficient private information retrieval. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E82-A(1):11–20, 1999.
- [32] T. Itoh. On lower bounds for the communication complexity of private information retrieval. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E84-A(1):157–164, 2001.
- [33] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proc. of the 32nd ACM Symp. on the Theory of Computing*, pages 80–86, 2000.
- [34] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *Proc. of the 35th ACM Symp. on the Theory of Computing*, pages 106–115, 2003.
- [35] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of the 38th IEEE Symp. on Foundations of Computer Science*, pages 364–373, 1997.
- [36] H. Lipmaa. An oblivious transfer protocol with log-squared communication. In J. Zhou and J. Lopez, editors, *the 8th Information Security Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328. Springer-Verlag, 2005.
- [37] E. Mann. Private access to distributed information. Master's thesis, Technion – Israel Institute of Technology, Haifa, 1998.
- [38] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. of the 33th ACM Symp. on the Theory of Computing*, 2001. Long version: Communication complexity and secure function evaluation, *Cryptology ePrint Archive*, number 2001/076, 2001.
- [39] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of the 31st ACM Symp. on the Theory of Computing*, pages 245–254, 1999.
- [40] K. Obata. Optimal lower bounds for 2-query locally decodable linear codes. In *RANDOM '02, 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, volume 2483 of *Lecture Notes in Computer Science*, pages 39 – 50. Springer-Verlag, 2002.
- [41] R. Ostrovsky and V. Shoup. Private information storage. In *Proc. of the 29th ACM Symp. on the Theory of Computing*, pages 294–303, 1997.
- [42] J. F. Raymond. Private information retrieval: Improved upper bound, extension and applications. Master's thesis, School of Computer Science, McGill University, 2000.
- [43] A. Razborov and S. Yekhanin. An $\Omega(n^{1/3})$ Lower Bound for Bilinear Group Based Private Information Retrieval. ECCC TR06-050, 2006. Available at <http://www.eccc.uni-trier.de/eccc>.
- [44] 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-Verlag, 1998.

- [45] L. Trevisan. Some applications of coding theory in computational complexity. *Quaderni di Matematica*, 13:347–424, 2004. Also appeared as ECCC TR04-043, 2004. Available at <http://www.eccc.uni-trier.de/eccc>.
- [46] S. Wehner and R. de Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In *Proc. of the 32nd International Colloquium on Automata, Languages and Programming*, pages 1424–1436, 2005.
- [47] D. Woodruff and S. Yekhanin. A geometric approach to information-theoretic private information retrieval. In *Proc. of the 20th IEEE Conf. on Computational Complexity*, pages 275–284, 2005.

A A High-Level View: Replication vs. Degree

In this appendix, we try to explain where the improvement over previous protocols comes from. The recursive formulation of our protocol, as well as the various technicalities involved in its implementation and analysis, make it somewhat difficult to trace the actual source for improvement. The following overview of our approach attempts to give a fairly accurate intuition as to *how* and *why* it works, while ignoring some details or difficulties that are dealt with in the technical sections.

A key technique in previous solutions, as well as in ours, is an *arithmetization* of the PIR problem. Using a polynomial representation for the database x , the user’s goal is reduced to evaluating a multivariate polynomial $P(\vec{Z})$, known to all k servers, on some point \vec{z} determined by its retrieval index i .¹⁰ This task should be achieved while hiding \vec{z} from each server. There are three parameters associated with the above problem: (1) the *number of variables* in P , denoted by m ; (2) the *degree* of P , denoted by d ; and (3) the *replication* amount k , i.e., the number of servers to which the polynomial P is known. The first two parameters determine the *description size* of P , which is $\Theta(m^d)$ when d is constant. Using an appropriate polynomial representation, the database size n is roughly equal to the description size of P , i.e. $n = \Theta(m^d)$.

A first observation is that if we manage to reduce the degree of P by a factor of c (without changing the number of variables m by much), the new description size will be roughly the c -th root of the original one. The new polynomial can be communicated to the user using only $O(n^{1/c})$ communication bits. We therefore try to reduce the degree of P , possibly updating the evaluation point \vec{z} , without revealing \vec{z} to any server. In what follows we first describe the previous approach to achieving this goal, following [3, 8], and then explain where we depart from this approach.

A degree reduction, as above, is achieved as follows. The user picks random points $\vec{y}_1, \dots, \vec{y}_k \in \mathbb{F}^m$ subject to the condition $\vec{y}_1 + \dots + \vec{y}_k = \vec{z}$, and sends each server \mathcal{S}_j a query consisting of all k points *except* \vec{y}_j . The intuition for this step is that it provides “maximal redundancy” subject to the requirement of hiding \vec{z} .¹¹ Note that the cost of this step is dominated by $m = O(n^{1/d})$. Thus, for the total communication complexity to be small, the initial representation degree d must be large.

The next step is to express the desired value $P(\vec{z})$ as the value of the polynomial $Q(\vec{Y}_1, \dots, \vec{Y}_k) \stackrel{\text{def}}{=} P(\vec{Y}_1 + \dots + \vec{Y}_k)$ at the point $(\vec{y}_1, \dots, \vec{y}_k)$. The advantage of switching to this new representation is that the value assigned to each of its variables is known to *almost all* servers (in contrast to the original polynomial $P(\vec{Z})$, whose evaluation point \vec{z} is completely unknown to the servers and should remain so).

¹⁰Here and in the following all polynomials are assumed to be over a finite field \mathbb{F} , which is taken to be $\text{GF}(2)$ by default.

¹¹Some justification for this intuition was recently given in [18]. It follows from [18] that shares of \vec{z} obtained as above can be *locally* converted by the servers into shares of \vec{z} from any other linear secret-sharing scheme (including the polynomial-based Shamir’s scheme). Thus, the above way for sharing \vec{z} between the servers is in some sense without loss of generality.

To make use of this advantage, we write Q as a sum of monomials. Every such monomial is a product of d variables. Each variable is missed by exactly one out of the k servers; hence, for the d variables involved in a given monomial there must be *at least one* server which misses *at most* d/k values of these variables. We now assign each monomial to one of the servers corresponding to this monomial, and let each server substitute specific values for all of the variables it knows in each of the monomials assigned to it. After this step, each server holds a polynomial P_j in its *unknown* variables \vec{Y}_j , such that the degree of P_j is at most d/k and $\sum_{j=1}^k P_j(\vec{y}_j) = P(\vec{z})$. Thus, we can complete the protocol by letting each server j send to the user a *description* of its lower-degree polynomial P_j (e.g., using a list of its coefficients), which allows the user to compute the desired value $P(\vec{z})$.

We now take a more quantitative look at the type of savings obtained by the above degree reduction technique. As discussed above, reducing the representation degree by a factor of k induces a $(1/k)$ -th power reduction in its size. By picking a “high” degree d , the queries sent to each server will be short, and the answers will be of length $O(n^{1/k})$. At a first glance, this seems to be the end of the road. However, a crucial observation is that the above degree analysis involves *integers*. Thus, the reduced degree is actually guaranteed to be bounded by $\lfloor d/k \rfloor$. While such integer truncation operations are typically viewed as a nuisance, in this case they turn out to make a big difference. In a sense, without this truncation effect a PIR protocol with $O(n^{1/k})$ communication is the best we would have.

How far can the advantage of truncation be pushed? Two useful examples are the following. First, assume that $d = k - 1$. In this case, we get the most evident benefit: $\lfloor d/k \rfloor = 0$, instead of $(k - 1)/k$ in the fractional case, implying that each polynomial P_j will have degree-0 (i.e., P_j will be a constant) and therefore can be described by one bit. However, a disadvantage of this choice of parameters is that d is rather small, and therefore the length of the queries will be rather large ($O(n^{1/d}) = O(n^{1/(k-1)})$). Still, a useful feature of the corresponding protocol is that it requires only one answer bit from each server. Indeed, the latter protocol was prior to this work essentially the best protocol of this type, yielding the best binary locally-decodable codes until this work. A second useful choice of parameters is $d = 2k - 1$. In this case the answers are longer than before: since $\lfloor d/k \rfloor = 1$ the degree is reduced by a factor of $d = 2k - 1$, and consequently the description length of P_j is $O(n^{1/d}) = O(n^{1/(2k-1)})$. However, since the queries now are also shorter, namely of length $O(n^{1/(2k-1)})$, we get a protocol of a smaller total communication complexity. The above protocol was the best known protocol (in terms of the total communication complexity) prior to this work.

In light of the above surprising effect of integer truncation on the complexity of PIR, it is natural to ask whether the savings can be pushed even further. We start with the following observation. The degree reduction process we used may be thought of as a way for *trading replication for degree*: We started with a polynomial P of degree d which is replicated among k servers, and ended up with polynomials P_j of degree $\lfloor d/k \rfloor$, each known to a *single* server. Thus, we have given away all of the original replication, and in return obtained the biggest possible gain in the degree. However, it is not clear a-priori that this greedy approach is optimal. An alternative approach that comes to mind is to apply several *partial* degree reduction steps, hoping to benefit multiple times from the integer truncation effect.

To this end, we generalize the above degree reduction procedure as follows. Suppose that we are willing to reduce the replication from k to k' (rather than to 1). Then, we may assign each monomial to some set V of k' servers which *jointly* miss the least number of variables from this monomial. This allows us to write the desired value $P(\vec{z})$ as the sum of values $Q_V(\vec{y})$, where each Q_V is a polynomial known to a set V of at least k' servers. Note that the maximal degree of Q_V increases as k' grows, and in any case is no more than $\lfloor dk'/k \rfloor$.

We return to the previous question: can we gain by reducing the degree (along with the replication) in

multiple steps? Intuitively, there is no advantage in applying the above “local” degree reduction process in multiple steps, as the final representation could have been directly attained in one step. A key idea that makes such a multi-step process useful is to use additional interaction with the user for adjusting the degree between each two reduction steps. In such a *degree conversion* step, both the number of variables and the degree are changed, but the description size remains the same. For instance, suppose that some set of k' servers holds a degree-3 polynomial Q in m variables, and the user holds some point \vec{z} whose additive shares $\vec{y}_1, \dots, \vec{y}_{k'}$ are replicated among the servers as above. Moreover, suppose that it is possible for the servers to locally compute a degree-2 polynomial Q' in $O(m^{3/2})$ variables and for the user to locally convert \vec{z} to a point \vec{z}' , such that $Q(\vec{z}) = Q'(\vec{z}')$. (Note that the existence of such a local conversion is plausible, since $(m^{3/2})^2 = m^3$, and so we have not decreased the description size.) Then, by re-sharing the point \vec{z}' among the servers, the user can adjust the degree to 2 without reducing the amount of replication or increasing the description size. Such a degree conversion procedure would allow to obtain additional savings by interleaving reduction steps with degree conversion steps.

We illustrate this by a 4-server example. Suppose that $d = 5$. Dispensing with all the replication in one step (i.e., by letting $k' = 1$), reduces the degree to $\lfloor 5/4 \rfloor = 1$. Instead, we let $k' = 3$. This brings the degree down to $\lfloor 5 \cdot 3/4 \rfloor = 3$, since for any monomial there is a set of 3 servers which jointly miss at most 3 variables from this monomial. Now, we adjust the degree to 2. This increases the number of variables to $O(m^{3/2}) = O((n^{1/5})^{3/2}) = O(n^{3/10})$, and requires the user to send additional queries of comparable size. Finally, we can apply the reduction step again to reduce the replication from 3 to 1. This brings the degree down to 0, and allows the servers to communicate $P(\vec{z})$ to the user by sending a single bit each. Thus, we obtain a protocol with query length $O(n^{3/10})$ and answer length 1 – improving the protocol with query length $O(n^{1/(k-1)})$ described above.

Unfortunately, we do not know whether the above degree conversion problem can be solved in general.¹² Instead, we get around this problem by relying on a specific *promise* on the value of the point \vec{z} held by the user, namely on the fact that \vec{z} is promised to be of a *constant weight*. Given this promise, we can realize the degree conversion via a recursive invocation of PIR (since evaluating a polynomial on a constant-weight assignment requires reading a constant number of coefficients). The main difficulty in implementing this approach is that the polynomials held by the servers following a degree reduction step should be evaluated at assignments \vec{y}_j that do not have a constant weight. Thus, the main technical challenge is to obtain good implementations of the degree reduction step in which the polynomials held by the servers only need to be evaluated at the constant-weight assignment \vec{z} (rather than its shares \vec{y}_j). The abstract linear algebra problem that corresponds to this challenge is described in Section 5, where we also describe a nearly optimal solution to this problem.

B Limitations of the Protocol

B.1 Limitations of the Concrete Protocol

In this section, we show that the analysis presented in Section 3.5 is optimal. This is not to say that there are no other protocols that can do better; it only says that within the freedom that our protocol has in choosing the parameters λ, k' , the choice we made that achieves complexity of $n^{O(\log \log k / (k \log k))}$ is essentially the best. More precisely, we analyze the recursion of Theorem 3.7 starting with any k -server protocol with

¹²Specifically, it is open if for every $d' < d$ it is possible to convert m -variate degree- d polynomials to m' -variate degree- d' polynomials where $m' = O(m^{d/d'})$. Such a conversion is possible whenever d' divides d , and is not known to be possible otherwise. The simplest open case is $d = 3, d' = 2$.

communication complexity of $O(n^{1/(ck)})$ for some constant c under the restriction that $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$. The last restriction is necessary for the current protocol.

Definition B.1 (Coefficient of a PIR protocol) We say that a PIR protocol \mathcal{P} has a coefficient $\alpha(k)$, where $\alpha : \mathbb{N} \rightarrow \mathbb{R}$, if

$$\forall k \ C_{\mathcal{P}}(n, k) = O_k(n^{1/(\alpha(k)k)}).$$

For example, the k -server protocol with complexity $O(n^{1/(2k-1)})$ has a coefficient $\alpha(k) = 2 - 1/k < 2$. The k -server protocol of Corollary 3.11 has a coefficient $\alpha(k) = \Theta(\log k / \log \log k)$.

Claim B.2 Suppose that we obtain a protocol \mathcal{P}' with a coefficient $\alpha_{\mathcal{P}'}$ from a protocol \mathcal{P} that has a coefficient $\alpha_{\mathcal{P}}$, using one step of recursion. Then, $\alpha_{\mathcal{P}'}(k) \leq \alpha_{\mathcal{P}}(k') + 1$.

Proof: Recall that $C_{\mathcal{P}'}(n, k) \geq n^{1/d} + C_{\mathcal{P}}(n^{\lambda k'/d}, k') \geq O_k(n^{1/d} + (n^{\lambda k'/d})^{1/(\alpha_{\mathcal{P}}(k')k')})$. Thus,

$$\alpha_{\mathcal{P}'}(k) \leq \min \left\{ \frac{d}{k}, \frac{d\alpha_{\mathcal{P}}(k')}{k\lambda} \right\}.$$

Furthermore, $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2) \leq (\lambda + 1)k$ (since $k' \geq 2$). Therefore,

$$\alpha_{\mathcal{P}'}(k) \leq \min \left\{ \lambda + 1, \alpha_{\mathcal{P}}(k') + \frac{\alpha_{\mathcal{P}}(k')}{\lambda} \right\}. \quad (12)$$

On one hand, if $\lambda \leq \alpha_{\mathcal{P}}(k')$ then $\alpha_{\mathcal{P}'}(k) \leq (\lambda + 1) \leq \alpha_{\mathcal{P}}(k') + 1$. On the other hand, if $\lambda > \alpha_{\mathcal{P}}(k')$ then $\alpha_{\mathcal{P}'}(k) \leq \alpha_{\mathcal{P}}(k') + \frac{\alpha_{\mathcal{P}}(k')}{\lambda} \leq \alpha_{\mathcal{P}}(k') + 1$. \square

Recall that in each recursion step we have $k' \leq k - 1$, and we use a k' -server scheme. Thus, the number of recursion steps is at most k . By Claim B.2, the coefficient is increased by at most 1 in each recursion step. This proves that the coefficient that can be achieved by our scheme is at most k . To prove a stronger bound, we show in the next claim that k' must be much smaller than k .

Claim B.3 If we use the recursion in a way that improves the complexity (that is, $C_{\mathcal{P}'}(n, k) \leq C_{\mathcal{P}}(n, k)$) then $k' \leq \frac{k}{\lambda - 1} + 2$.

Proof: By Equation (4), the protocol uses a recursion with a database of size $n^{\lambda(k-1)/d}$ and $k - 1$ servers. If $\lambda(k - 1) \geq d$ then this is worse than the original problem with database of size n and k servers. Thus, $\lambda(k - 1) < d$. Recall that $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$. Therefore, $\lambda(k - 1) < (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$ which implies the claim. \square

Claim B.4 Assume that we use the recursion starting with any k -server protocol with coefficient c for some constant c . Under the restriction that $d \leq (\lambda + 1)k - (\lambda - 1)k' + (\lambda - 2)$, the resulting protocol has a coefficient no larger than $\alpha(k) = O(\log k / \log \log k)$.

Proof: Let ℓ be the number of recursion steps and $k_{\ell} = k$. We construct a k_{ℓ} -server protocol \mathcal{P}_{ℓ} using recursion with some parameters k' and λ . In particular, this implies that we use some k' -server protocol $\mathcal{P}_{\ell-1}$. This proceeds recursively where, in each stage, we construct a k_i -server protocol \mathcal{P}_i using recursion with some parameters k'_i and λ_i and, in particular, we use some k'_i -server protocol \mathcal{P}_{i-1} . Using this notation, $k_{i-1} = k'_i$. This recursion ends after some finite number $\ell - 1$ of steps, where we use a k_1 -server protocol

with coefficient c . Let \mathcal{P}_i be a protocol with a coefficient $\alpha_i(k)$, and let $\beta_i = \alpha_i(k_{i-1}) - 2$ for $i \in \{2, \dots, \ell\}$. By possibly stopping the analysis before the recursion ends, we can assume, without loss of generality, that $\alpha_i(k_{i-1}) \geq 4$ for every $i \in \{2, \dots, \ell\}$ (and, therefore, $\beta_i \geq 2$) and that $k_1 \geq 3$.

By Equation (12), $\beta_i \leq \lambda_i - 1$ and, by Claim B.3,

$$k_{i-1} \leq \frac{k_i}{\lambda_i - 1} + 2 \leq \frac{k_i}{\beta_i} + 2.$$

Thus, $k_i \geq k_{i-1}\beta_i - 2\beta_i$. By induction,

$$k_\ell \geq k_1 \prod_{i=2}^{\ell} \beta_i - 2 \sum_{j=1}^{\ell-1} \prod_{i=\ell-j+1}^{\ell} \beta_i.$$

Since $\beta_i \geq 2$ for every i , it holds that

$$\prod_{i=\ell-j+1}^{\ell} \beta_i \leq \frac{\prod_{i=2}^{\ell} \beta_i}{2^{\ell-j}}.$$

Therefore,

$$\sum_{j=1}^{\ell-1} \prod_{i=\ell-j+1}^{\ell} \beta_i \leq \left(\sum_{j=1}^{\ell-1} 2^j \right) \left(2^{-\ell} \prod_{i=1}^{\ell} \beta_i \right) \leq \prod_{i=1}^{\ell} \beta_i,$$

and

$$k_\ell \geq (k_1 - 2) \prod_{i=2}^{\ell} \beta_i \geq \prod_{i=2}^{\ell} \beta_i$$

(the last inequality holds since $k_1 \geq 3$). By Claim B.2, for every integer s , where $\beta_2 \leq s \leq \beta_\ell$ there is an i such that $s \leq \beta_i < s + 1$. Thus, $k_\ell \geq (\beta_\ell - 1)!/\beta_2!$. Notice that, since c is a constant, then β_2 is a constant. Recall that $k_\ell = k$, $\beta_\ell = \alpha_\ell(k_{\ell-1}) - 2$; thus, the coefficient of every protocol that can be constructed by our recursion is at most $\alpha_\ell(k) = O(\log k / \log \log k)$, as claimed. \square

B.2 Limitations of the Abstract Protocol

In this section we show that the abstract framework, presented in Section 5, cannot yield a protocol with communication complexity better than $O(n^{1/(3k \log k)})$. That is, while there is some hope to improve the complexity of our protocol using the abstract framework, the possible improvement is not dramatic.

The proof is by induction on k . Assume that for every protocol \mathcal{P} , used in the recursion, it holds that $C_{\mathcal{P}}(n, k) = \Omega(n^{1/(3k \log k)})$. Recall that, given a protocol \mathcal{P} and a spanning block set \mathcal{B} , the complexity of the resulting protocol \mathcal{P}' is $O(n^{1/d} + \sum_{b \in \mathcal{B}} C_{\mathcal{P}}(n^{\delta(b)/d}, |V(b)|))$. Fix d and k such that $d \geq 3k \log k$ (if $d < 3k \log k$ there is nothing to prove). Consider a monomial in which each server appears at least $\lfloor d/k \rfloor$ times and at most $\lceil d/k \rceil$ times. This monomial has to be covered by at least one block $b \in \mathcal{B}$. Let $v \stackrel{\text{def}}{=} |V(b)|$ and $\delta \stackrel{\text{def}}{=} \delta(b)$. Thus,

$$C_{\mathcal{P}'}(n, k) \geq C_{\mathcal{P}}(n^{\delta/d}, v) = \Omega(n^{\delta/(d \cdot 3v \log v)}).$$

To complete the proof, it is enough to bound $\delta/(d \cdot 3v \log v)$.

First assume that $v \leq k/2$. In such block b , it must hold that

$$\delta \geq \lfloor d/k \rfloor v \geq (d/k - 1)v. \tag{13}$$

Thus,

$$\begin{aligned}
\frac{\delta}{d \cdot 3v \log v} &\geq \frac{d/k - 1}{3d \log v} \\
&\geq \frac{d/k - 1}{3d \log(k/2)} = \frac{1 - k/d}{3k \log(k/2)} \\
&\geq \frac{1 - k/(3k \log k)}{3k \log(k/2)} \geq \frac{\log k - 1}{3k \log k \log(k/2)} \\
&\geq \frac{\log(k/2)}{3k \log k \log(k/2)} = \frac{1}{3k \log k}.
\end{aligned}$$

Now assume that $k/2 < v \leq k$. For this case, we need a better bound on δ than the bound given in Equation (13).¹³

Claim B.5

$$\delta \geq \frac{vd}{k} - \frac{(k-v)v}{k}.$$

Proof: Recall that we consider a monomial in which each server appears at least $\lfloor d/k \rfloor$ times and at most $\lceil d/k \rceil$ times and that we want to cover the monomial with a block in which v servers do not appear. We will show that each set of v servers appears at least $\frac{vd}{k} - \frac{(k-v)v}{k}$ times in this monomial.

Let $\beta \stackrel{\text{def}}{=} d \bmod k$. That is, β servers appear $\lfloor d/k \rfloor + 1$ times in the monomial and $k - \beta$ servers appear $\lfloor d/k \rfloor$ times in the monomial. If $v \leq k - \beta$, then the number of times every set of v servers appears in the monomial is at least $v \lfloor d/k \rfloor = v(d - \beta)/k \geq v(d - k + v)/k$ as promised. If $v > k - \beta$, then the number of times every v servers appear in the monomial is at least

$$v \lfloor d/k \rfloor + v - (k - \beta) = v(d - \beta)/k + v - k + \beta.$$

This expression increases when β increases and $\beta > k - v$. Thus, also in this case the number of times every v servers appear in the monomial is at least as stated in the claim. \square

We next bound $\delta/(d \cdot 3v \log v)$ for $k/2 < v \leq k$.

$$\begin{aligned}
\frac{\delta}{d \cdot 3v \log v} &\geq \frac{vd}{kd \cdot 3v \log v} - \frac{(k-v)v}{kd \cdot 3v \log v} \\
&\geq \frac{1}{3k \log v} - \frac{k-v}{9k^2 \log k \log v} \\
&= \frac{1}{3k \log k} \frac{3k \log k - (k-v)}{3k \log v}.
\end{aligned}$$

To complete the proof, we need to show that

$$\frac{3k \log k - (k-v)}{3k \log v} \geq 1.$$

That is, we need to show that

$$3 \frac{k}{v} \log \frac{k}{v} - \frac{k}{v} + 1 \geq 0.$$

¹³This bound holds also for the case $v \leq k/2$, but the cruder bound is sufficient there.

By elementary calculus, $\log x \geq x - 1$ for $1 \leq x \leq 2$. Thus,

$$3\frac{k}{v} \log \frac{k}{v} - \frac{k}{v} + 1 \geq 3\frac{k}{v} \left(\frac{k}{v} - 1 \right) - \frac{k}{v} + 1.$$

Recall that $v \leq k$. Thus, we define the function $f(x) \stackrel{\text{def}}{=} 3x(x-1) - x + 1 = 3x^2 - 4x + 1$, and prove that $f(x) \geq 0$ for $x \geq 1$. Note that $f(1) = 0$ and the minimum of the function $f(x)$ is for $x = \frac{2}{3} < 1$, thus $f(x)$ is an increasing function when $x \geq 1$ and therefore positive. This completes the proof.