Improved Upper Bounds on Information-Theoretic Private Information Retrieval

(extended abstract)

Yuval Ishai* Eyal Kushilevitz †

Abstract

Private Information Retrieval (PIR) schemes allow a user to retrieve the i-th bit of an n-bit database x, replicated in k servers, while keeping the value of i private from each server. A t-*private* PIR scheme protects the user's privacy from any collusion of up to t servers. The main cost measure for such schemes is their communication complexity.

We introduce a new technique for the construction of information-theoretic (i.e., unconditionally secure) PIR schemes, providing a non-trivial linear-algebraic generalization of previous techniques. Using this technique, we improve and simplify known upper bounds on the communication complexity of PIR schemes in the information-theoretic setting. In the case of 1-private PIR, we give a simple k-server scheme with complexity $O(k^3 n^{1/(2k-1)})$, improving the best known construction whose complexity also grows linearly in $n^{1/(2k-1)}$ for any fixed k, but depends *exponentially* on k. Our improvements are more significant for t-private PIR schemes, where $t > 1$. For example, we get a 2-private, 4-server PIR scheme whose communication complexity is $O(n^{1/3})$, compared to the previously known $O(n^{1/2})$ upper bound.

1 Introduction

Private Information Retrieval (PIR) schemes allow a user to retrieve information from a database, while keeping the identity of this information secret from the server in which the database is stored. It is convenient to model the database by an n-bit string x and the information that the user is interested in retrieving from x is modeled as the i-th bit of x; that is, x_i, for some index i. The notion of PIR was introduced by Chor et. al [9] and since then was the subject of a significant amount of work [2, 6, 7, 8, 11, 12, 13, 15, 17]. If information-theoretic privacy is required and there is a single server available, then there is no better solution other than the user asking for a copy of the whole database from the server (which is obviously very inefficient) [9]. However, if identical copies of the database are replicated in $k > 1$ servers, then the communication complexity can be significantly reduced. In particular, for every *constant* k, there exists a k-server PIR scheme with communication complexity $O(n^{1/(2k-1)})$ (this result was first shown in [9] for the case $k = 2$; this 2-server scheme was later used as a building block in [2] to prove the result for $k > 2$). More precisely, the complexity of the k-server scheme in [2] is $O(2^{k^2} n^{1/(2k-1)})$. When $k = \Theta(\log n)$ a communication complexity of $O(\log^2 n \log\log n)$ is attainable (this was shown in [9] using a scheme similar to the *instance hiding* schemes of [3, 4]). Finally, [15] show that under the quadratic-residuosity assumption single-server *computationally-private* schemes of communication complexity $O(n^\epsilon)$, for any $\epsilon > 0$, are possible. Extensions of this result to other "standard" intractability assumptions appear in [16], and an efficiency improvement based on a new intractability assumption appears in [7]; also in the multi-server model computationally-private schemes were studied in [8]. The current work is concerned with the information-theoretic setting.[1]

There are several parameters according to which a multi-server PIR scheme is evaluated. Most importantly, k, the number of servers (copies) is the most crucial resource; then, the communication complexity of the scheme; finally, a *privacy threshold* parameter t ($1 \le t \le k$) which limits the number of servers that might collude together in order to get information regarding the index i that the user is interested in retrieving. In all of the abovementioned results, it is assumed that no two servers can talk to each other;

*Department of Computer Science, Technion, Haifa 32000, Israel. E-mail: yuvali@cs.technion.ac.il. Part of this work was done while visiting IBM T.J. Watson Research Center.

†IBM T.J. Watson Research Center, and Department of Computer Science, Technion. E-mail: eyalk@watson.ibm.com and eyalk@cs.technion.ac.il. Supported in part by the Mitchell-Schoref program at the Technion.

[1] Information-theoretic PIR schemes have several advantages over the currently-known computationally-private schemes: (1) the time complexity of the information-theoretic schemes is much smaller; (2) for "practical" parameters of k and n they even give smaller communication complexity; and (3) to-date, there is no single-server low-communication PIR scheme which is based on a "general" intractability assumption.

that is, $t = 1$. The case $t > 1$ is addressed in [9], where it is shown that for any positive integers t, d there exists a t-private $(dt - t + 1)$-server PIR scheme with complexity $O(n^{1/d})$. Such t-private schemes are obtained by using a variant of the low-degree polynomial interpolation technique of [3, 4], along with a generic communication balancing technique due to [9]. Moreover, the communication complexity of the t-private schemes from [3, 4, 9] becomes poly-logarithmic when $k = \Theta(t \cdot \log n / \log\log n)$.

Since the first works on information-theoretic PIR, no progress has been made towards improving the upper bounds mentioned above.

Our results. In order to present our new improved upper bounds in a clear way, we give an alternative formulation of the question: instead of asking what is the best communication complexity that we can achieve for t-private PIR with k servers, we ask:

> Given values t and d, what is the smallest number of servers, k, for which we can design t-private scheme with communication complexity $O(n^{1/d})$?

We present a linear-algebraic approach that generalizes in a non-trivial way the cubes technique and the server-emulation technique of [9]. Using this approach, we construct schemes with various parameters. These include:

- For the case $t = 1$, we obtain a 1-private k-server PIR scheme with communication complexity $O(k^3 n^{1/(2k-1)})$. While for a *constant* k this bound is asymptotically the same as the previously known result for the case $t = 1$ [2], our scheme has two advantages. First, it is somewhat simpler (in particular, the scheme from [2] is recursive). Second, and more importantly, the communication complexity of the scheme from [2] depends exponentially on k; hence, our exact communication complexity is substantially better, starting from small values of k. The fact that the communication in our scheme grows polynomially in k allows us to get a poly-logarithmic communication complexity when k is $O(\log n / \log\log n)$. This yields some improvements over known *locally random reduction* and *instance hiding* schemes [1, 3, 4].

- For the case $t > 1$ our improvements are much more significant. We are able to reduce the number of servers required for obtaining t-private schemes with communication complexity $O(n^{1/d})$ for various parameters t, d (note that even reducing the number of servers by 1 might be considered a significant savings). For example, prior to our work the communication complexity of the best known 2-private scheme using $k = 4$ servers was $O(n^{1/2})$, while our bounds imply a scheme with communication complexity $O(n^{1/3})$. (Any further improvement on this communication complexity would immediately imply the same improvement for the 1-private, 2-server case.) More generally, for any odd[2] constant $d \geq 3$ and any constant $t > 1$, there exists a t-private k-server PIR scheme with communication complexity $O(n^{1/d})$, for $k = \min\left(\left\lfloor dt - \frac{d+t-3}{2} \right\rfloor, dt - t\right)$. We note that depending on the relation between d and t the minimum is given by a different expression and that in any case we save at least one server comparing to the previously known bound of $dt - t + 1$.

[2]In the case where d is even the bounds are almost the same.

k	t	new	previous
k	1	$O(k^3 n^{1/(2k-1)})$	$O(2^{k^2} n^{1/(2k-1)})$ [2]
4	2	$O(n^{1/3})$	$O(n^{1/2})$ [9]
6	2	$O(n^{1/4})$	$O(n^{1/3})$ [9]
12	2	$O(n^{1/8})$	$O(n^{1/6})$ [9]
6	3	$O(n^{1/3})$	$O(n^{1/2})$ [9]

Figure 1: communication complexity for some values of k, t

The communication complexity of our schemes for some values of k, t is summarized in Figure 1.

A natural question is how far these upper bounds can be pushed. We use the linear algebraic structure of our schemes to prove some lower bounds on the communication complexity of schemes which follow our paradigm. We emphasize that this only shows a lower bound on the power of our technique; for non-restricted PIR schemes no good lower bounds are known: the best lower bound known is $(4 - o(1)) \cdot \log n$ bits (compared to the trivial $\log n$ lower bound) and even this requires some non-trivial argument [16].

Techniques used. Our schemes employ a replication-based secret-sharing scheme due to [14]. Similarly to [5], we exploit some homomorphism property of this scheme (a variant of a multiplication property that was studied for general linear secret-sharing schemes in [10]). However, and quite interestingly, in the works mentioned above the sole motivation for using such replication-based schemes is that of dealing with general (i.e., non-threshold) access structures, whereas our work is only concerned with the threshold case.

Extensions. To keep the presentation as simple as possible, we focus on the basic notion of PIR as defined in [9]. However, our techniques can be used to derive improved upper bounds for various extensions of PIR: t-private retrieval of multi-bit records [9], schemes that provide "privacy" for the database in addition to the user [13], and schemes that keep the data itself private from the servers [12]. Details of these extensions are omitted from the current version.

Organization: In Section 2 we provide some basic notations and definitions that are used throughout the paper. In Section 3 we deal with the case $t = 1$. This case already demonstrates many of our ideas. In Section 4 we generalize our technique to handle efficiently larger values of t. Section 5 includes a lower bound on the power of our technique. Finally, Section 6 includes some concluding remarks and suggestions for future research.

2 Preliminaries

2.1 Notation

We use the following notation. For any positive integers k and t, we denote by $[k]$ the set of integers $\{1, \ldots, k\}$, and by $\binom{[k]}{t}$ the set $\{T \subseteq [k] : |T| = t\}$. We use e_j to denote a unit vector (that is, a vector with 1 in its j-th coordinate and 0 in all others), whose underlying vector space will be clear from the context. For $x, y \in F^n$, where F is a finite field, we let $\langle x, y\rangle$ denote the standard inner product over F; that is, $\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i$ (where both addition and multiplication are carried over F).

2.2 Definitions

A *private information retrieval* (PIR) scheme is a protocol for a user, $\mathcal{U}$, and k servers $\mathcal{DB}_1, \ldots, \mathcal{DB}_k$. Each server holds an identical copy of an n-bit string x which is called the database. The user has an index i which it is interested in retrieving. For this, the user chooses a random string r and based on i and r it computes a k-tuple $q_1, \ldots, q_k$ of queries. It sends each query q_j to the corresponding server $\mathcal{DB}_j$. Then, each server $\mathcal{DB}_j$ based on the database x and the query q_j computes an answer a_j. Finally, the user based on i, r and $a_1, \ldots, a_k$ computes some output b. We say that the scheme is correct if for all choices of x, i and r the output b equals x_i. We say that the scheme is t-private if for every set $T = \{j_1, \ldots, j_t\}$ of t servers, for every two indices i, i', and for every t-tuple of queries $q_{j_1}, \ldots, q_{j_t}$ we have

$$Pr(q_{j_1}, \ldots, q_{j_t} | i) = Pr(q_{j_1}, \ldots, q_{j_t} | i'),$$

where the probability is over the choice of r. [3]

The *communication complexity* of a PIR scheme is the total number of bits exchanged between the user and the k servers in the worst case. That is

$$\max_{x,i,r} \left(\sum_{j=1}^{k} |q_j| + \sum_{j=1}^{k} |a_j| \right).$$

We sometimes refer separately to the *query complexity* and to the *answer complexity*. These are naturally defined in the same manner as above.

3 Constructing 1-private PIR Schemes

In this section we deal with the case of 1-private PIR schemes. We start (in Section 3.1) by presenting an algebraic framework (generalizing the "cubes" approach of [9]), which will be used by all PIR schemes constructed in this work. Then (in Subsection 3.2), we present a basic scheme that utilizes this algebraic view. Finally (in Subsection 3.3), we optimize our approach and get our improved upper bounds for the case $t = 1$.

[3] The definition can be weakened by allowing multi-round schemes and introducing a probability of error. However, all the schemes presented in this work are one-round and always correct.

3.1 Algebraic Framework

In this section we define some algebraic constructs which are used throughout this work. As in [9], we fix a positive integer d (the *dimension*), and assume without loss of generality that $n = \ell^d$ for some positive integer ℓ. Let F be a finite field. The parameters n, ℓ and F induce the vector spaces $X \stackrel{\text{def}}{=} F^n$, called the *data space*, and $Q \stackrel{\text{def}}{=} F^\ell$, called the *query space*. In the following, the data string x will be viewed as a vector in the data space, and the queries sent by the user will consist of several vectors in the query space.

Let γ be a 1-1 mapping of $[n]$ onto $[\ell]^d$. For instance, $\gamma(i)$ may be taken as the base-ℓ representation of i. Finally, letting $\gamma(i) = (i_1, \ldots, i_d)$, we define a d-argument function $\prod : Q^d \to X$, whose i-th entry $(1 \le i \le n)$ is given by:

$$\left[\prod((q_1^1, \ldots, q_\ell^1), \ldots, (q_1^d, \ldots, q_\ell^d))\right]_i \stackrel{\text{def}}{=} q_{i_1}^1 \cdot q_{i_2}^2 \cdots q_{i_d}^d .$$

Notice that each i-th entry of the output n-tuple of $\prod$ is a product of a single entry from each input ℓ-tuple, where the index i determines (using γ) which d entries are selected. We will heavily rely on the following two (easily-verifiable) properties of $\prod$:

1. For every $i \in [n]$, if $\gamma(i) = (i_1, \ldots, i_d)$, then
$$\prod(e_{i_1}, \ldots, e_{i_d}) = e_i$$
2. $\prod$ is a *multilinear function* over Q. That is, for any $h \in [d]$ and $q^1, \ldots, q^d, \hat{q}^h \in Q$,
$$\begin{aligned}\prod(q^1, \ldots, q^{h-1}, q^h + \hat{q}^h, q^{h+1}, \ldots, q^d) \\ = \prod(q^1, \ldots, q^{h-1}, q^h, q^{h+1}, \ldots, q^d) \\ + \prod(q^1, \ldots, q^{h-1}, \hat{q}^h, q^{h+1}, \ldots, q^d) .\end{aligned}$$

Throughout the remainder of this paper, unless otherwise specified, F will be taken to be $\mathbf{GF}(2)$. When $F = \mathbf{GF}(2)$, the above algebraic constructs are isomorphic to the subcube geometry used in [9]. [4]

3.2 The Basic Approach

All PIR schemes constructed in this work conform to the following *linear* paradigm:

- The user represents i as $\gamma(i) = (i_1, \ldots, i_d)$, and shares each of the d ℓ-tuple e_{i_h} among the k servers using a t-private linear secret-sharing scheme;
- Each server performs a local computation on its shares, resulting in a collection of vectors from the data space X, and returns to $\mathcal{U}$ the inner product of the database x with each of these vectors;
- $\mathcal{U}$ reconstructs x_i by taking a fixed linear combination (depending only on i) of the answers.

[4] It will be later discussed (see Remark 1 in Section 4) why larger fields F should be considered even when retrieving a single bit.

We now explain in more detail how the scheme proceeds, for the case $t = 1$. Let $\gamma(i) = (i_1, \ldots, i_d)$. The first step in the scheme will be for the user to produce an *additive* secret-sharing of each unit vector e_{i_h} into m shares. That is, for each $1 \leq h \leq d$, the user chooses uniformly at random $q_h^1, \ldots, q_h^m$ such that $q_h^1 + q_h^2 + \ldots + q_h^m = e_{i_h}$. (Equivalently, $q_h^1, \ldots, q_h^{m-1}$ may be picked independently at random, and q_h^m may be set as $e_{i_h} - \sum_{s=1}^{m-1} q_h^s$). The key observation is that, by the properties of $\prod$ and the multilinearity of the inner-product, we can write:

$$\begin{aligned} x_i &= \langle x \ , \ e_i \rangle \\ &= \left\langle x \ , \ \prod(e_{i_1}, \ldots, e_{i_d}) \right\rangle \\ &= \left\langle x \ , \ \prod\left(\sum_{s=1}^{m} q_1^s, \ldots, \sum_{s=1}^{m} q_d^s\right) \right\rangle \\ &= \left\langle x \ , \ \sum_{a_1, \ldots, a_d \in [m]} \prod(q_1^{a_1}, \ldots, q_d^{a_d}) \right\rangle \\ &= \sum_{a_1, \ldots, a_d \in [m]} \left\langle x \ , \ \prod(q_1^{a_1}, \ldots, q_d^{a_d}) \right\rangle . \end{aligned}$$

Each expression $\langle x, \prod(q_1^{a_1}, \ldots, q_d^{a_d}) \rangle$ in the last sum will be referred to as a *term*. Notice that each term evaluates to a single field element (a single bit when $F = \mathbf{GF}(2)$).

At this point, we have not yet specified what the value of m is and how the user distributes the shares among the servers. However, the intuition provided by the above equation is that if a single server holds the shares $q_1^{a_1}, \ldots, q_d^{a_d}$, then this server will be able to compute the value of the term $\langle x, \prod(q_1^{a_1}, \ldots, q_d^{a_d}) \rangle$. Moreover, if for each of the terms there is a server that can compute it, we can assign each term to one server, and given such assignment construct a PIR scheme as follows: (1) each server will compute the *sum* of all terms assigned to it and send this partial sum to the user; (2) by adding all k partial sums the user will reconstruct x_i. In light of the above, a natural idea that comes to mind is to use secret-sharing with *replication*; that is, each share will be given to many servers, and this will intuitively increase the likelihood that the shares required for computing a particular term are all known to a single server. Similarly to [5] (though for a different purpose) we use the replication-based secret-sharing scheme of [14].

We first describe a simple implementation which does not yield the best obtainable communication complexity but provides an integration of the above ideas. For this, we choose $m = k = d + 1$ (where k is the number of servers). For each i_h, the user shares e_{i_h} into $q_h^1, \ldots, q_h^k$ as above, and distributes the share q_h^a to all the servers *except* for $\mathcal{DB}_a$. Notice that each server views all shares but one, which is sufficient to guarantee the privacy of e_{i_h} on one hand, and on the other hand will provide the servers with a valuable resource of redundancy. The only observation that is left to make in this simple case is that since each term involves d shares and the total number of servers (or shares) is $d + 1$, then for every d-tuple $(q_1^{a_1}, \ldots, q_d^{a_d})$ there exists a server which received all d shares, and therefore can compute the value of the corresponding term.

The communication in the above scheme consists of $d^2 \cdot \ell = d^2 n^{1/d}$ bits sent from the user to each server, and a single bit replied from each server (a standard communication balancing technique [9] can be applied to reduce the complexity to the order of $n^{1/(d+1)}$, which is still much worse than the scheme we present next).

3.3 More Powerful Covers (still for $t = 1$)

In this subsection we show a way of covering all terms using almost half as many servers, providing a clean generalization of the 2-server scheme from [9].

Assume $k > d/2$. As before, we let the user additively share each e_{i_h} as $q_h^1 + \ldots + q_h^k$, and distribute each share q_h^a to all servers except $\mathcal{DB}_a$. Now, consider a term $\tau = \left\langle x \ , \ \prod(q_1^{a_1}, \ldots, q_d^{a_d}) \right\rangle$. If there exists some $j \in [k] \backslash \{a_1, \ldots, a_d\}$ then, as before, τ can be computed by the server $\mathcal{DB}_j$ alone. Unfortunately, it might also be the case that no server holds *all* shares $q_1^{a_1}, \ldots, q_d^{a_d}$, as the index set $\{a_1, a_2, \ldots, a_d\}$ may cover the entire server domain $[k]$. However, since $d < 2k$, in such a case there must exist some server $\mathcal{DB}_j$ which misses exactly one share q_h^j, where $a_h = j$. (If all servers miss at least two shares, then each index in $[k]$ occurs at least twice in the d-tuple $(a_1, a_2, \ldots, a_d)$, implying that $d \geq 2k$). The term τ will be *emulated* by server $\mathcal{DB}_j$ as follows. Knowing all shares of e_{i_h} except q_h^j, server $\mathcal{DB}_j$ is able to construct an ℓ-bit list L_τ which includes the value of τ as its i_h-th entry. Specifically, the p-th bit in the list L_τ, $1 \leq p \leq \ell$, can be calculated as:

$$\left\langle x \ , \ \prod\left(q_1^{a_1}, \ldots, q_{h-1}^{a_{h-1}}, e_p - \sum_{a \in [k] \backslash j} q_h^a, q_{h+1}^{a_{h+1}}, \ldots, q_d^{a_d}\right) \right\rangle .$$

Notice that since the position i_h is known to $\mathcal{U}$, it can extract the value of τ from the list L_τ.

As a final optimization observe that instead of sending separately all lists L_τ for the terms it is assigned to emulate, each server may add up (coordinate-wise) all the lists with the same emulation position h, $1 \leq h \leq d$, resulting in d ℓ-bit lists. Moreover, the bit value of the terms it is assigned to evaluate directly may be added to all elements of one list. Thus, it suffices for each server to reply with $d\ell$ bits. Again, the user who knows i_h can extract the corresponding value from the combined list. To conclude, the scheme described above shows the following:

Theorem 1. *Let $k \geq 2$, and $d = 2k - 1$. Then, there exists a 1-private k-server PIR scheme with communication complexity of $O(k^3 n^{1/d}) = O(k^3 n^{1/2k-1})$ bits (more precisely, the query complexity is $k(k-1)dn^{1/d}$ bits and the answer complexity is $kdn^{1/d}$ bits).*

3.4 Application to Instance Hiding

Notice that the k-server bound in Theorem 1 is polynomial in k, in contrast to the k-server scheme of [2] whose communication complexity is exponential in k. In particular, for $k = \Omega(\frac{\log n}{\log \log n})$, the communication becomes polylogarithmic in n. This allows pushing the applicability of

Theorem 1 to the instance hiding scenario, where the bounded player's computation, let alone the communication with the oracles, is required to be polynomial in the input length (or poly-logarithmic in the corresponding "database" size).

Corollary 1. *Let $f_m : \{0,1\}^m \to \{0,1\}$ be any family of Boolean functions. Then, for any constant $c > 0$, there exists a non-adaptive $(\frac{m}{c\log m})$-oracle instance hiding scheme in which the bounded player's computation (and communication) complexity is $\tilde{O}(m^{c/2+3})$, where $\tilde{O}(f) \stackrel{\text{def}}{=} O(f \cdot \log^{O(1)}(f))$.*

Proof. Viewing the evaluation of f_m as retrieval from a 2^m-bit data string (namely, the truth-table of f_m), the specified communication bound is obtained from the bound in Theorem 1 by letting $n = 2^m$ and $k = \frac{m}{c\log m}$. Indeed, in that case

$$\begin{aligned} O(k^3 n^{1/d}) &= O\left((\frac{m}{c\log m})^3 \cdot (2^m)^{\left(\frac{2m}{c\log m}-1\right)^{-1}} \right) \\ &= \tilde{O}(m^3 \cdot (2^m)^{\frac{c\log m}{2m}}) \\ &= \tilde{O}(m^{c/2+3}). \end{aligned}$$

Furthermore, the k-server scheme described in this section can be implemented such that the user's computation is linear in the communication.[5] □

The bound in Corollary 1 should be compared with the corresponding bound for the schemes in [4, 9], which is $\tilde{O}(m^{c+2})$.

4 General Privacy Thresholds

In this section we generalize and extend the technique described in the previous section to construct t-private PIR schemes with larger privacy thresholds t.

4.1 The Paradigm

Fix parameters t, d, k, and let $\ell = n^{1/d}$. Later we will consider the question of how large k should be, as a function of t and d, so that a PIR scheme corresponding to these parameters can actually be constructed.

As before, the user represents i as $\gamma(i) = (i_1, \ldots, i_d)$, where each $i_h \in [\ell]$, and shares each unit vector e_{i_h} among the servers. The 1-private secret-sharing scheme of the previous section is generalized to the following replication-based t-private scheme. First, each unit vector e_{i_h} is *additively* shared into $m = \binom{k}{t}$ shares, labeled by t-subsets of $[k]$. That is, $e_{i_h} = \sum_{T\in\binom{[k]}{t}} q_h^T$, where the shares q_h^T are random subject to the above restriction. Then, $\mathcal{U}$ sends to each server $\mathcal{DB}_j$ all shares q_T^h such that $j \notin T$. Notice that the joint view of any collusion $S \subseteq [k]$ of t servers will be uniformly random over the query domain, since they will miss one share of each secret (namely, the share q_h^S of each secret i_h). Now, similarly to the previous section,

$$\begin{aligned} x_i &= \langle x \ , \ e_i \rangle \\ &= \left\langle x \ , \ \prod(e_{i_1}, \ldots, e_{i_d}) \right\rangle \\ &= \left\langle x \ , \ \prod(\sum_{T\in\binom{[k]}{t}} q_1^T, \ldots, \sum_{T\in\binom{[k]}{t}} q_d^T) \right\rangle, \end{aligned}$$

which may be decomposed into

$$\sum_{T_1,\ldots,T_d\in\binom{[k]}{t}} \left\langle x \ , \ \prod(q_1^{T_1}, \ldots, q_d^{T_d}) \right\rangle.$$

From now on, the term $\langle x, \prod(q_1^{T_1}, \ldots, q_d^{T_d})\rangle$ will be denoted by the d-tuple of sets $(T_1, \ldots, T_d)$. As before, the user's goal is to learn the sum of all terms. It would clearly suffice for the user to learn any set of *linear combinations* of terms, as long as this set *spans* the above sum of all terms.

Now, what linear combinations of terms are known to, or can be efficiently emulated by, a particular server $\mathcal{DB}_j$? Generalizing observations made in the previous section, server $\mathcal{DB}_j$ can:

1. compute any single term $(T_1, \ldots, T_d)$ such that $j \notin \bigcup_{h\in[d]} T_h$;
2. *emulate* any sum of terms of the form

$$\sum_{T\in\binom{[k]}{t}} (T_1, \ldots, T_{h-1}, T, T_{h+1}, \ldots, T_d)$$

   such that $j \notin \bigcup_{h'\in[d]\setminus\{h\}} T_{h'}$, by generating a list L of length ℓ whose i_h-th position includes the sum value. Specifically, the bit in the p-th position in this list is defined by:

$$L_p = \left\langle x \ , \ \prod(q_1^{T_1}, \ldots, q_{h-1}^{T_{h-1}}, e_p, q_{h+1}^{T_{h+1}}, \ldots, q_d^{T_d}) \right\rangle.$$

   (The idea is that the sum of all shares in the h-th coordinate of the terms equals some unit vector e_p; the list contains this sum for all ℓ possibilities.)

We refer to the first kind of linear combination as a valid *0-block*, and to the second kind as a valid *1-block* (corresponding to the dimension of the set of terms it involves).[6] For brevity, the 1-block

$$\sum_{T\in\binom{[k]}{t}} (T_1, \ldots, T_{h-1}, T, T_{h+1}, \ldots, T_d)$$

will be denoted by the d-tuple

$$(T_1, \ldots, T_{h-1}, *, T_{h+1}, \ldots, T_d),$$

[5] Observe that by letting ℓ be a power of 2, $\gamma(i)$ is trivial to compute from the binary representation of i.

[6] In principle, blocks of higher dimensions can be used as well. However, the cost of their emulation will usually be prohibitive for our purposes. See Section 6 for some more discussion.

where the wildcard "*" indicates summation over all possible substitutions. We stress that not any d-tuple of t-sets represents a valid 0-block, and not any d-tuple with a single wildcard represents a valid 1-block. For instance, $(\{2,4\},*,\{1,3\})$ is not a valid 1-block when $k=4$, since $\{2,4\}$ and $\{1,3\}$ cover the set $[4]$, but it *is* a valid 1-block when $k=5$.

We now define two useful notions related to the expressive power of a set of blocks in a given term space.

Definition 1. *Fix t,d,k. Let $\mathcal{T} \subseteq \binom{[k]}{t}^d$ be a set of terms and $\mathcal{B}$ be a set of blocks. We say that $\mathcal{B}$* covers *$\mathcal{T}$, if any term $\tau \in \mathcal{T}$ is included in some block from $\mathcal{B}$. We say that $\mathcal{B}$* spans *the sum of terms in $\mathcal{T}$ (or* spans *$\mathcal{T}$ for short), if there exists a linear combination of the blocks in $\mathcal{B}$ resulting in the sum $\sum_{\tau\in\mathcal{T}}\tau$ (where each block is viewed as a linear combination, with coefficients from F, of terms from the entire space). Finally, we say that a linear combination (or a set) of terms is* spanned, *if it is spanned by the set of all valid 0-blocks and 1-blocks.*

Remark 1. (On the role of the field F) *Notice that whether or not a given set of blocks spans a given set of terms may depend on the underlying field F. For instance, if there exists a set of blocks such that every term is covered by* exactly *4 blocks from this set, then the sum of all terms is clearly spanned over* $\mathbf{GF}(3)$, *but is not necessarily spanned over* $\mathbf{GF}(2)$. *Nevertheless, we use $F=\mathbf{GF}(2)$ throughout this section since this is sufficient (and at one point necessary) for our upper bounds. Our lower bounds however apply to any field F.*

As outlined by the previous discussion, spanning the sum of all terms using 0-blocks and 1-blocks has direct application to the construction of efficient PIR schemes. Specifically, a straightforward generalization of the scheme outlined in Section 3, together with the above ideas, gives:

Theorem 2. *Let d,t,k be positive integers, such that the sum of all terms in the term space $\binom{[k]}{t}^d$ is spanned by the valid 0-blocks and 1-blocks in this space. Then, there exists a t-private k-server PIR scheme with query complexity $k\binom{k-1}{t}dn^{1/d}$ and answer complexity $kdn^{1/d}$ (where $\binom{k-1}{t}$ is the number of shares assigned to each server).*

4.2 Spanning the Sum of all Terms

Theorem 2 motivates studying the following question:

Question 1. *Given d and t, how large should k be so that the set of valid 0-blocks and 1-blocks spans the sum of all terms?*

In this subsection we will solely focus on this question. First, we introduce some notation.

Definition 2. *Let $k_c(d,t)$ denote the minimal integer k such that the term space $\binom{[k]}{t}^d$ is* covered *by its 0-blocks and 1-blocks,[7] and $k_s(d,t)$ denote the minimal k such that the sum of all terms is* spanned *by these blocks.*

[7] Since each 0-block is covered by (exactly d) 1-blocks, it is sufficient in the definition of k_c to consider 1-blocks.

An exact characterization of $k_c(d,t)$ and some simple bounds on $k_s(d,t)$ are given by the following two claims.

Claim 1. *For any positive integers d and t, $k_c(d,t)$ is the minimal integer greater than $dt/2$. That is,*

$$k_c(d,t) = \lfloor dt/2\rfloor + 1.$$

Proof. We show that term space $\binom{[k]}{t}^d$ is covered by 1-blocks if and only if $k > dt/2$. If $dt \geq 2k$, then it is possible to construct a term $\tau = (T_1,T_2,\ldots,T_d)$ such that each element $j\in[k]$ occurs in at least two sets T_h (e.g., by greedily adding each element to a pair of vacant sets) and such term cannot be covered by a 1-block. Conversely, if there exists a term $\tau=(T_1,T_2,\ldots,T_d)$ which is not covered by any 1-block, then every element $j\in[k]$ must occur in at least two sets T_h, implying that $dt\geq 2k$. □

Claim 2. *For any positive integers d and t,*

$$k_c(d,t) \leq k_s(d,t) \leq dt+1.$$

Proof. The fact that k_c is a lower bound on k_s is an immediate consequence of the definitions. The upper bound on $k_s(d,t)$ follows from the fact that the *0-blocks* alone are sufficient to cover the entire term space, and hence span the sum of all terms, whenever $k > dt$ (generalizing the observation made in Subsection 3.2 for the case $t=1$). □

The scheme in the previous section shows that when $t=1$, $k_s(d,t) = k_c(d,t) = \lfloor dt/2\rfloor+1$. That is, the lower bound on k_s provided by Claim 2 is tight for the case $t=1$. However, this is not the case in general (see Section 5). In the following we will improve the upper bound on $k_s(d,t)$ for $t>1$. A first observation to make is that, in addition to the valid 0-blocks, there are other singleton sets of terms which can be spanned.

Lemma 1. *If a term $\tau=(T_1,T_2,\ldots,T_d)$ includes a set T_h which is* disjoint *from all $d-1$ other sets, then $\{\tau\}$ is spanned.*

Proof. Let $\tau=(T_1,T_2,\ldots,T_d)$ be such a term, with T_h satisfying $T_h\cap T_{h'}=\emptyset$ for any $h'\in[d]\setminus\{h\}$. The definition of valid blocks implies that:

- $(T_1,\ldots,T_{h-1},*,T_{h+1},\ldots,T_d)$ denotes a valid 1-block (since the $d-1$ sets in this d-tuple do not include any element of T_h, and thus cannot cover $[k]$); and
- For any $T\in\binom{[k]}{t}\setminus\{T_h\}$, the d-tuple

  $$(T_1,\ldots,T_{h-1},T,T_{h+1},\ldots,T_d)$$

  denotes a valid 0-block (since $T_h\setminus T\neq\emptyset$ and each element of $T_h\setminus T$ is not covered by the d sets).

Finally, writing τ as:

$$\begin{aligned}\tau &= (T_1,\ldots,T_{h-1},*,T_{h+1},\ldots,T_d)\\ &- \sum_{T\in\binom{[k]}{t}\setminus\{T_h\}}(T_1,\ldots,T_{h-1},T,T_{h+1},\ldots,T_d),\end{aligned}$$

the lemma follows. $\square$

We can now focus our attention on the terms which we do not know how to handle directly.

Definition 3. *A term $(T_1, T_2, \ldots, T_d)$ is said to be* bad*, if it is not a 0-block, and it does not include a set T_j disjoint from all others.*

Lemma 2. *Suppose that a term set $\mathcal{T}$ containing all bad terms (and possibly some other terms) can be spanned. Then the set of all terms can be spanned.*

Proof. The 0-blocks and the terms handled by Lemma 1 (replaced by their spanning sets of blocks) span $\binom{[k]}{t}^d \setminus \mathcal{T}$. $\square$

We are now ready to prove upper bounds on $k_s(d,t)$. The first bound applies only to *odd* d. The second bound is slightly weaker, but can be applied also when d is even.

Claim 3. *For any positive integer $t \geq 1$ and* odd *integer $d \geq 3$,*

$$k_s(d,t) \leq \min\left(\left\lfloor dt - \frac{d+t-3}{2}\right\rfloor \ , \ dt - t\right).$$

Proof. It suffices to show that when d is odd, and when either $k > dt - \frac{d+t-1}{2}$ or $k > dt - t - 1$, then the sum of all terms is spanned. Suppose k, d and t meet the above constraints. By Lemma 2, it suffices to show that some term set $\mathcal{T}$ containing all bad terms can be spanned. We now show that this is achieved by the block set $\mathcal{B}$ which includes all 1-blocks in which the element "1" occurs an *even* number of times.

Lemma 3. *For k, d, t constrained as above, each bad term is included in an odd number of blocks from the set $\mathcal{B}$.*

Proof. Suppose that this is not the case; that is, there exists a bad term $\tau = (T_1, \ldots, T_d)$ which is covered by an even number of blocks from $\mathcal{B}$. Now, consider the number of occurrences of '1' in τ.

- If '1' does not occur at all, then τ is not bad, contradicting the assumptions.
- If '1' occurs exactly once, in the set T_h, then there is exactly one block from $\mathcal{B}$ covering τ. Specifically, the block obtained by "removing" T_h (i.e., replacing it with a wildcard) contains an even number of 1's (0), and it is a valid 1-block since its sets do not contain '1'. We conclude that this case as well contradicts the assumptions.
- If '1' occurs e times, where e is *even*: there are $(d-e)$ *candidates* for blocks in $\mathcal{B}$ covering τ, namely all those obtained by removing a set which does not contain '1'. Since d is odd and e is even, $d-e$ is odd. It follows that for τ to be covered by an even number of blocks from $\mathcal{B}$, at least one of the $d-e$ candidates must not be a valid 1-block, implying that some set T_h *which does not contain '1'* must be covered by all other sets.
- If '1' occurs o times, where o is odd and is greater than 1, then there are o candidates for blocks in $\mathcal{B}$ covering τ, namely all those obtained by removing a set which contains '1'. Again, it follows that for τ to be covered by an even number of blocks from $\mathcal{B}$ at least one of these candidates must not be a valid 1-block, implying that at least one of the (at least three) sets T_h which contain '1' must be covered by all other sets.

We deduce from the above cases that there must exist sets T_z, T_{p_1} and T_{p_2}, with *distinct* indices z, p_1, p_2, such that: (a) T_z is covered by the union of all other $d-1$ sets; and (b) $1 \in T_{p_1} \cap T_{p_2}$. Moreover, since τ is bad, we have: (c) every set T_h intersects some other set; and (d) the union of all d sets is $[k]$.

We now ask the following extremal question: What is the maximal k such that the above 4 conditions can be simultaneously satisfied by some term τ? Equivalently, what is the maximum size of the *union* of d sets $T_1, \ldots, T_d$ of size t each, such that conditions (a),(b), and (c) above are met? Denote this maximum union size by $k^*_{odd}(d,t)$. The following two propositions bound $k^*_{odd}(d,t)$ from above.

Proposition 1. $k^*_{odd}(d,t) \leq dt - t - 1$.

Proof. For any sets $T_1, \ldots, T_d$ meeting the above conditions (a) and (b), we have:

$$\begin{aligned}\left|\bigcup_{h\in[d]} T_h\right| &= \left|\bigcup_{h\in[d]\setminus\{z\}} T_h\right| \\ &\leq (d-1)t - 1 \\ &= dt - t - 1,\end{aligned}$$

where the first equality follows from condition (a) and the inequality from condition (b). $\square$

Proposition 2. $k^*_{odd}(d,t) \leq dt - \frac{d+t-1}{2}$.

Proof. Suppose that $T_1, \ldots, T_d$ meet the above conditions (a) and (c). For each $w \in T_z$, let h_w be an index of some set other than T_z which includes w (where the indices h_w need not be distinct, and their existence is guaranteed by condition (a)). Let $Z = \{z\} \cup \{h_w : w \in T_z\}$. Next, for every $h \in [d] \setminus Z$, we select an element y_h which occurs both in T_h and in some other set (again, the y_h need not be distinct, and their existence is guaranteed by condition (c)). Finally, we cluster the sets with indices from $[d] \setminus Z$ according to their selected y_h. That is, for each $w \in [k]$ we define a "cluster" $G_w \stackrel{\text{def}}{=} \{h \in [d] \setminus Z : y_h = w\}$.

We analyze how the union size grows, when we first take all sets with indices from Z, then add each cluster G_w containing two or more sets, and finally add all clusters G_w containing a single set. The union of all sets from Z is no larger than $t|Z| - t$, since each element of T_z is counted there at least twice. When adding a cluster G_w with at least two sets, its $|G_w|$ sets contribute at most $t|G_w| - (|G_w| - 1) \leq (t - \frac{1}{2})|G_w|$ new elements to the union size. Finally, we add together all singleton clusters G_w. Suppose there are s of them. Then, their total contribution to the union size is no larger than $(t-1)s$, since the element y_h in each such unmatched set T_h either occurs in a previously added set, or occurs in some other singleton cluster (and will only be accounted for once). Altogether, we have that:

$$\begin{aligned}\left|\bigcup_{h\in[d]} T_h\right| &\leq \left|\bigcup_{h\in Z} T_h\right| + \left|\bigcup_{h\in[d]\setminus Z} T_h\right| \\ &\leq (t|Z|-t) + (t-1/2)(d-|Z|) \\ &= dt - t + (|Z|-d)/2 \\ &\leq dt - t + (t+1-d)/2 \\ &= dt - (d+t-1)/2,\end{aligned}$$

as required. □

We remark that the above analysis is tight; that is, k^*_{odd} can be shown to be *equal* to $\min(dt-t-1, \lfloor dt - \frac{d+t-1}{2}\rfloor)$.

It may now be concluded that when d is odd and $k > k^*_{odd}(d,t)$, no bad term can be covered by an even number of blocks from $\mathcal{B}$; combining this with Propositions 1 and 2, Lemma 3 follows. □

Since all linear combinations are taken over $F = \mathbf{GF}(2)$, Lemma 3 implies that a set containing all bad terms can be spanned, which by Lemma 2 concludes the proof of Claim 3. □

We now prove a slightly weaker bound on $k_s(d,t)$, which applies to *any* d (and in particular to any even d). In fact, when $d \geq t+2$ this bound is identical to the previous one, and otherwise the difference between the bounds is 1.

Claim 4. *For any positive integers d, t,*

$$k_s(d,t) \leq \min\left(\left\lfloor dt - \frac{d+t-3}{2}\right\rfloor \ , \ dt - t + 1\right).$$

Proof. The proof goes along the lines of the proof of Claim 3. Again, we define a set of blocks $\mathcal{B}$, which is guaranteed to span some set containing all bad blocks whenever $k \geq k_s(d,t)$. In this case, we let $\mathcal{B}$ be the set of all valid 1-blocks of the form $(*, T_2, \ldots, T_d)$.

As before, suppose that $\tau = (T_1, \ldots, T_d)$ is a bad term which is covered by an even number of blocks from $\mathcal{B}$. Then, the set T_1 must clearly be covered by the union of the other sets, implying that condition (a) from the proof of Claim 3 holds with $h = 1$. Since τ is assumed to be bad, conditions (c) and (d) apply as well. Defining k^*_{even} analogously to k^*_{odd}, the proof of Proposition 2 can be used to show that $k^*_{even}(d,t) \leq dt - \frac{d+t-1}{2}$ (since only conditions (a),(c) and (d) are used in this proof). Finally, $k^*_{even}(d,t) \leq dt - t$, since for any sets $T_1, \ldots, T_d$ meeting condition (a) we have:

$$\left|\bigcup_{h\in[d]} T_h\right| = \left|\bigcup_{h\in[d]\setminus\{1\}} T_h\right| \leq dt - t.$$

□

Substituting the bounds from Claims 3 and 4 in Theorem 2, we conclude the following:

Corollary 2. *For any constant integers $d \geq 2$ and $t \geq 1$, there exists a t-private k-server PIR scheme with communication complexity $O(n^{1/d})$, where*

$$k = \min\left(\left\lfloor dt - \frac{d+t-3}{2}\right\rfloor \ , \ dt - t + 1 - (d \ mod \ 2)\right).$$

More precisely, the complexity of our scheme with parameters d, t and k as above is $O(k^2\binom{k}{t} \cdot n^{1/d})$.

Notice that when $t \geq d-1$, our bound improves the t-private schemes of [9] only for an odd d (in which case a single server is being saved). When d is large compared to t, the savings becomes much more substantial. For instance, fixing $t = 2$, the number of servers required for achieving $O(n^{1/d})$ communication drops from the previously known bound of $2d-1$ to $\lceil 3d/2 \rceil$.

5 A Lower Bound on k_s

In this section we present a *tight* lower bound on $k_s(d,t)$ for the case $d = 3$. The most significant implication of this bound is a strong separation between the cover bound k_c and the span bound k_s over *any* field F.

Claim 5. *For any $t > 1$, and over any field F,*

$$k_s(3,t) \geq 2t.$$

Proof. We show that with $k = 2t-1$ the sum of all terms is not spanned. Construct the following list of $2t-3$ terms:

$$\begin{gathered}(A, B_0, C_0)\\(A, B_1, C_0)\\(A, B_1, C_1)\\(A, B_2, C_1)\\(A, B_2, C_2)\\ \vdots \\(A, B_{t-2}, C_{t-3})\\(A, B_{t-2}, C_{t-2})\end{gathered}$$

where to define the sets $A, \{B_j\}, \{C_j\}$, consider the "cycle" of sets

$$A - B_0 - C_0 - B_1 - C_1 - \ldots - B_{t-2} - C_{t-2} - A.$$

This is a cycle of odd length $(2t-1)$, each edge of which represents a pair of sets which we would like to cover $[k]$. We define the sets in this cycle as follows: each set will be taken to include t (cyclically) consecutive elements, where the "first" element in each set is the last element in the previous one. (Notice that the number of sets in the cycle suffices exactly for the first set to be consecutive to the last one.) For instance, for $t = 5$ and $k = 2t-1 = 9$ the sets will be: $A = \{1,2,3,4,5\}, B_0 = \{5,6,7,8,9\}, C_0 = \{9,1,2,3,4\}, B_1 = \{4,5,6,7,8\}, C_1 = \{8,9,1,2,3\}, B_2 = \{3,4,5,6,7\}, C_2 = \{7,8,9,1,2\}, B_3 = \{2,3,4,5,6\}, C_3 = \{6,7,8,9,1\}$.

From the way that the sets and the terms were constructed it follows that:

- None of the terms in the above list is a 0-block (since $B_j \cup C_j = B_j \cup C_{j-1} = [k]$).
- Any 1-block either covers no term, or covers two consecutive terms in the list (since there is no 1-block of the form $(*, B, C)$ that covers any of the above terms and each 1-block of the form $(A, B_j, *)$ or $(A, *, C_j)$ covers two terms).

Now, assign to the terms with an odd position in the list the weight 1, to the terms with an even position in the list weight -1, and to all remaining terms in the term space the weight 0. It may be concluded from the above that each block has weight 0, and hence each linear combination of blocks has weight 0 as well. Since the sum of the weights of all terms is nonzero (as the list is of odd length), the sum of all terms cannot be spanned. $\square$

We note that the lower bound of Claim 5 is tight, since it matches the upper bound of Claim 3. Finally, combining the lower bound on k_s provided by Claim 5 with the exact characterization of k_c in Claim 1, we obtain the desired separation of k_s from k_c.

6 Concluding Remarks

We present a new general framework for the construction of information-theoretic PIR schemes, and obtain improved and simplified schemes based on this framework. We would like to point out the following two open problems, related to possible improvements of the results of this work:

- Is $k > dt/2$ sufficient for the existence of t-private k-server PIR schemes of complexity $O(n^{1/d})$? This bound on k coincides with our cover bound k_c, and is the best one could hope for without improving the 2-server $O(n^{1/3})$ bound from [9]. We have ruled out the sufficiency of our technique (and some of its obvious generalizations) for obtaining such a result. However, it may still be the case that a different approach can be used to construct such schemes.
- Can the k^3 factor in our $O(k^3 n^{1/(2k-1)})$ upper bound for 1-private k-server PIR schemes be improved? More generally, can one avoid the $\binom{k}{t}$ multiplicative overhead induced by our use of replication-based secret-sharing (e.g., by using more efficient threshold secret-sharing schemes)?

Finally, there are several directions for extending and further generalizing our technique. Below we mention two directions which are related to our interpretation of the scheme from [2] in our framework. One such possible direction is to exploit blocks of higher dimension (i.e., not only 0-blocks and 1-blocks as we currently use). The extra emulation cost of such a block might be handled if it is replicated in sufficiently many servers, or if the dimension of the term space is increased proportionally to the block's dimension.

Another direction is to use other secret-sharing schemes for generating the queries. We have some evidence for the usefulness of this second approach; namely, using a different secret-sharing scheme, it is possible to obtain a *1-private* PIR scheme whose communication complexity is roughly half the complexity of the corresponding 1-private scheme from Section 3. This scheme, which may be viewed as an optimized version of the scheme from [2], is sketched in Appendix A. We leave open the question whether this example only indicates a slight redundancy in the amount of replication we use in our schemes, or in fact different secret-sharing schemes can yield more substantial improvements.

Acknowledgment

We thank Niv Gilboa for very helpful discussions.

References

[1] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle (extended abstract). In *Proc. of 19th STOC*, pages 195–203, 1987. Journal version in JCSS vol 39 pp. 21-50, 1989.

[2] A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *Proc. of 24th ICALP*, volume 1256 of *Lecture Notes in Computer Science*, pages 401–407, 1997.

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

[4] 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, LNCS 537, pages 62-76.

[5] D. Beaver and A. Wool. Quorum-based secure multi-party computation. In *Proc. of EUROCRYPT'98, LNCS 1403, Springer Verlag*, pages 375–390, 1998.

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

[7] C. Cachin, S.Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology - EUROCRYPT '99*, 1999.

[8] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of 29th STOC*, pages 304–313, 1997.

[9] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of 36th FOCS*, pages 41–50, 1995. To appear in JACM.

[10] R. Cramer, I. Damgard, and U. Maurer. Span programs and secure general multiparty computation. BRICS Report Series, RS-97-28, 1997. Available from http://www.brics.dk.

[11] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for database private information retrieval. In *Proc. of the 17th Annu. ACM Symp. on Principles of Distributed Computing*, pages 91–100, 1998.

[12] Y. Gertner, S. Goldwasser, and T. Malkin. A random server model for private information retrieval (or how to achieve information theoretic PIR avoiding data replication). In *Proc. of 2nd RANDOM*, 1998.

[13] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. of the 30th Annu. ACM Symp. on the Theory of Computing*, pages 151–160, 1998.

[14] M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structures. In *Proc. IEEE Global Telecommunication Conf., Globecom 87*, pages 99–102, 1987.

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

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

[17] R. Ostrovsky and V. Shoup. Private information storage. In *Proc. of 29th STOC*, pages 294–303, 1997.

[18] A. Shamir. How to share a secret. *Commun. ACM*, 22(6):612–613, June 1979.

A An Alternative 1-Private Scheme

In this appendix we sketch the details of an alternative to the 1-private scheme from Section 3. This scheme, also discussed in Section 6, may be viewed as an optimized adaptation of the scheme from [2] to our framework.

Fix k, and let $d = 2k-1$. The user first splits each secret unit vector e_{i_h}, $1 \leq h \leq d$, into two additive shares, $k-1$ times independently. That is, e_{i_h} is written as

$$e_{i_h} = a_1^h + b_1^h = a_2^h + b_2^h = \cdots = a_{k-1}^h + b_{k-1}^h,$$

where the shares a_j^h, b_j^h are random subject to the above restriction. Then, each share a_j^h is sent to the server $\mathcal{DB}_j$ alone, and each share b_j^h is sent to all servers $\mathcal{DB}_{j'}$ with $j' > j$. Under such secret-sharing, a term (or a 0-block) is represented by a d-tuple over the symbols $a_1, b_1, a_2, b_2, \ldots, a_{k-1}, b_{k-1}$, and a d'-block is represented by such d-tuple with d' entries replaced by a wildcard. As the next claim shows, there exists a linear combination of valid 0-blocks and 1-blocks yielding the desired d-block $(*, *, \ldots, *)$.[8]

Claim 6. *For every integer z, $0 \leq z \leq k$, any block β such that:*

1. *the dimension of β is at most $d - 2z$*
2. *all (non-"$*$") entries of β are from the set $\{b_1, \ldots, b_z\}$*

is spanned by valid 0-blocks and 1-blocks.

Proof. The proof proceeds by descending induction on z. When $z = k$, then β must be either a 0-block or a 1-block containing only b-type shares, and hence can be directly emulated by $\mathcal{DB}_k$. Now, let $z < k$. We assume that the claim holds for $z+1$ and prove that it holds for z. Without loss of generality, consider a d'-block

$$\beta = (*, *, \ldots, *, s_{d'+1}, \ldots, s_d),$$

where $d' \leq d - 2z$ and all entries s_j are from the set $\{b_1, \ldots, b_z\}$. Expressing "$*$" as "$(a_{z+1} + b_{z+1})$", we can write β as the following sum of terms:

$$\beta = \sum_{\tau \in \{a_{z+1}, b_{z+1}\}^{d'} s_{d'+1} \cdots s_d} \tau.$$

It remains to show that the above sum of terms can be spanned. We partition the set of terms in the above sum into ones that contain at most one occurrence of b_{z+1} and ones that contain at least two such occurrences. The sum of the former type of terms can be directly spanned using the 0-blocks and 1-blocks held by $\mathcal{DB}_{z+1}$. To see that terms of the latter type can also be spanned by valid blocks, consider (without loss of generality) the term

$$\tau = (a_{z+1}, \ldots, a_{z+1}, b_{z+1}, b_{z+1}, s_{d'+1}, \ldots, s_d).$$

Expressing "a_{z+1}" as "$(* - b_{z+1})$", the term τ can be written as a linear combination of blocks, where each such block is over the symbols $\{b_1, \ldots, b_{z+1}\}$ and its dimension is at most $d - 2z - 2 = d - 2(z+1)$. By the induction assumption, all such blocks can be spanned. □

For the special case $z = 0$, we conclude from the above claim that the d-block $(*, *, \ldots, *)$ is spanned by valid 0-blocks and 1-blocks. This allows the servers and the user to proceed according to the paradigm described in Sections 3, 4.

[8] This notion of spanning the d-block $(*, *, \ldots, *)$ may be defined as a natural generalization of the notion of spanning the sum of all terms from Section 4. For secret-sharing schemes as above, however, there may exist many different linear combinations of terms yielding the same block.