Private Information Retrieval*

Benny Chor† Oded Goldreich‡ Eyal Kushilevitz§ Madhu Sudan¶

April 21, 1998

Abstract

Publicly accessible databases are an indispensable resource for retrieving up to date information. But they also pose a significant risk to the privacy of the user, since a curious database operator can follow the user's queries and infer what the user is after. Indeed, in cases where the users' intentions are to be kept secret, users are often cautious about accessing the database. It can be shown that when accessing a single database, to completely guarantee the privacy of the user, the whole database should be down-loaded; namely n bits should be communicated (where n is the number of bits in the database).

In this work, we investigate whether by replicating the database, more efficient solutions to the private retrieval problem can be obtained. We describe schemes that enable a user to access k replicated copies of a database ($k \geq 2$) and *privately* retrieve information stored in the database. This means that each individual server (holding a replicated copy of the database) gets no information on the identity of the item retrieved by the user. Our schemes use the replication to gain substantial saving. In particular, we present a two-server scheme with communication complexity $O(n^{1/3})$.

*A preliminary version of this paper appeared in *Proc. of 36th IEEE Conference on the Foundations of Computer Science (FOCS)*, pp. 41-50, October 1995.

†Computer Science Dept., Technion, Haifa, Israel. Email: `benny@cs.technion.ac.il`

‡Computer Science and Applied Math. Dept., Weizmann Institute of Science, Rehovot, Israel. Supported by grant No. 92-00226 from the Israel–US Binational Science Foundation (BSF), Jerusalem, Israel. Email: `oded@wisdom.weizmann.ac.il`

§Computer Science Dept., Technion, Haifa, Israel. Email: `eyalk@cs.technion.ac.il` http://www.cs.technion.ac.il/~eyalk

¶Laboratory for Computer Science, MIT, 545 Technology Sq., Cambridge, MA 02139, USA. E-mail: `madhu@theory.lcs.mit.edu`. Some of this work was done when the author was at IBM.

1 Introduction

Consider a user that makes a query to a database. A lot of research was devoted to methods that protect the database against a "curious" user. For example, there are methods that enable a user to ask queries to a *statistical database* in a way that prevents him from reconstructing the value of particular entities (e.g., [2, 9, 13, 14, 26] and [27, Section 10.5]).

It may seem surprising at first glance that there are no methods to protect the privacy of the *user*. For example, an investor that queries the stock-market database for the value of a certain stock, may wish to keep private the identity of the stock he is interested in. However, it is not difficult to prove (see Section 5.1) that if the user wants to keep its privacy (in the information theoretic sense), then essentially the only thing he can do is to ask for a copy of the *whole database*. Clearly, this is too much communication overhead, which makes it practically unacceptable.

The rapid development of distributed databases (see [8]) and fast communication networks results in many scenarios in which the same database is replicated at several sites. This raises hope to get around the difficulty of achieving privacy in the single server scenario. It may be possible to make queries to several servers such that from the answers the desired information can be obtained, while each server (by observing only the query sent to him) gets no information on the identity of the item the user is interested in.

Before going any further let us make the problem more concrete. We view the database as a binary string $x = x_1 \cdots x_n$ of length n. Identical copies of this string are stored by $k \geq 2$ servers. The user has some index i, and he is interested in obtaining the value of the bit x_i. To achieve this goal, the user queries each of the servers and gets replies from which the desired bit x_i can be computed. The query to each server is distributed independently of i and therefore each server gains no information about i. A scheme with these properties is called a Private Information Retrieval (PIR) scheme.

We present various PIR schemes with significantly smaller communication complexity than the obvious n-bit solution (i.e., asking for a copy of x). In particular, we obtain a two-server scheme with communication complexity $O(n^{1/3})$. Our schemes are based on exclusive-or (linear summations, or `sum`) queries; this type of queries is very common and is actually implemented in several "real-world" databases (see [9, 14, 27]).

1.1 Omitted from this Version

Our original work [12] contained a full description of

- Schemes for a constant number, k, of servers with communication complexity $O(n^{1/k})$.
- A scheme for $\frac{1}{3}\log_2 n + 1$ servers with total communication complexity $\frac{1}{3}(1 + o(1)) \cdot \log_2^2 n \cdot \log_2 \log_2(2n)$.

These constructions are based on polynomial interpolation. They are similar to (but more efficient than) schemes presented in [5, 6] for the related (but different) context of *instance hiding* (see discussion below). Furthermore, these schemes have been subsumed by subsequent work of Ambainis [3]. Following a recommendation by an anonymous referee, the description of these schemes was omitted from the current version.

Also omitted with the abovementioned schemes are their modifications to a setting in which privacy is maintained with respect to coalitions of $t > 1$ servers. In particular, for integer function

t and integer constant $c > 1$, a $t(n)$-private information retrieval scheme for $c \cdot t(n)$ servers, with communication complexity $O(t(n) \cdot \sqrt[c]{n})$, was given.

1.2 Related Work

For the case $k = 2$ (i.e., two servers), the question whether replication of databases can help was explicitly asked, but *not* answered, by Fortnow and Szegedy [15]. A first indication that something better than the user asking for a copy of x can be done is given by a result of Pudlák and Rödl [22]. With a complexity-theory motivation in mind they studied the following question. There are three players: player S_1 that holds a string x and an index j, player S_2 that holds the same string x and an index ℓ, and player U that knows both j and ℓ. The goal is for S_1 and S_2 to send a single message each to U so that he will be able to compute the bit $x_{j+\ell \bmod n}$. Pudlák and Rödl show that this can be done using $o(n)$ bits (more precisely, $O(n \log\log n / \log n)$). Using their protocol, a two-server PIR scheme can be constructed as follows: The user chooses uniformly at random a pair of indices j, ℓ such that $j + \ell = i \bmod n$. He sends j to the first server, ℓ to the second and the three of them execute the [22] protocol. This yields a two-server PIR with communication complexity $o(n)$.

Independently of our work, Babai, Kimmel, and Lokam [4] studied the following problem, related to the one studied in [22] (where, again, the motivation comes from complexity theory). There are $k+1$ players $S_1, \ldots, S_k$ and U. The player U holds k indices $i_1, \ldots, i_k$ (each is an ℓ bit string). Each player S_j holds an $n = 2^\ell$ bit string x (common to all of them) and all the indices but i_j. The goal is for each S_j to send a single message to U so that U will know the value of the bit $x_{i_1 \oplus i_2 \oplus \cdots \oplus i_k}$, where $\oplus$ here denotes bitwise exclusive-or. A protocol for this problem can be transformed into a private information retrieval scheme with an additional cost of $k(k-1)\log_2 n$ bits as follows: The user chooses uniformly at random k indices ($\log_2 n$ bit strings) $i_1, \ldots, i_k$ such that $i_1 \oplus \cdots \oplus i_k = i$. He then sends to the j-th server all the indices but i_j, and the servers execute the protocol. Babai et al. [4] obtain the following results: for $2 \leq k < \log_2 n$ players the total communication is $O(kn^{H_2(1/(k+1))})$ (where $H_2(\cdot)$ is the Binary Entropy function), and for $k \geq \log_2 n$ the total communication is $2\log_2 n$. For example, for $k = 2$ their protocol (and hence the private information retrieval scheme) uses $O(n^{H_2(1/3)}) \approx O(n^{0.92})$ bits. For $k = c\log_2 n$ the communication is polylogarithmic (note however, that the transformation into a private information retrieval scheme will cost additional $c^2 \log_2^3 n$ bits in this case). To conclude, using the results of [4] one can get much better private information retrieval schemes than those that can be obtained using [22], but still not as good as the schemes constructed in our paper.

In [25, 1, 24, 5, 6] the *instance hiding* problem is introduced and studied. In this problem, a computationally bounded player U that holds an input (instance) i wishes to compute a known function f on input i. The function f may be hard to compute, so U can query k computationally unbounded oracles to achieve this task (each oracle can compute $f(j)$ for any j). Still, the player wants to keep its input i hidden from the oracles. In a sense, this problem can be viewed as if the oracles have a string $f(1)f(2)\ldots f(n)$ and U wants to obtain the i^{th} bit of this string, which is the value $f(i)$, while keeping i private. In this sense the instance hiding model is related to the model of private information retrieval. Some of the techniques used in [5, 6] are relevant to our problem, especially the use of low degree polynomials, introduced by Beaver and Feigenbaum [5], and further developed by Beaver, Feigenbaum, Kilian and Rogaway [6]. We remark that the results of [5, 6] cannot be directly applied to the important case of two servers. For details see [12].

It should be emphasized that despite these similarities, there are substantial differences between the models and between the quality of the results. In our model the value n is considered a feasible quantity, while in the instance hiding model n is exponential in the length of the instance, so it is an infeasible quantity. Consequently, the instance-hiding model is aimed towards $poly(|i|)$-time computations for U, allowing only solutions in which the communication between the user and the servers is poly-logarithmic in n. In contrast, the main thrust of our work is the case with small number, k, of servers (specifically, a constant like 2). We do allow the user to perform n^ε time computation (where $\varepsilon > 0$ is a constant), and in particular send and receive messages longer than polylog(n).

1.3 Subsequent Work

Following the conference presentation of this work, Ambainis [3] constructed more efficient Private Information Retrieval (PIR) schemes for $k > 2$ servers. His k-server scheme achieves communication complexity $O(n^{1/(2k-1)})$. The construction is recursive: A $k+1$-server scheme is built from a given k-server scheme (with certain properties). The basis of the recursion (i.e., $k = 2$) is our two-servers scheme.

Ostrovsky and Shoup have extended the PIR scope, and invented schemes for *private information storage* [21]. These are schemes which, in the same distributed scenario as PIR, enable users both to read and to write into the database in a private manner (where privacy in this case is only with respect to the servers and *not* against other users). Interestingly, Ostrovsky and Shoup achieve this with an addition of one server and a poly-logarithmic communication overhead (compared to the retrieval only schemes). They use and adapt techniques of Oblivious RAM [18], and inherit some properties of this construction. In particular, the protocols are multi-round and the data is stored in coded form (in particular, different servers do *not* hold replications of the same data). We stress that [21] is the only work to date which utilizes multi-round protocols.

Chor and Gilboa [10] have relaxed the privacy requirement by considering computationally-bounded servers. This weaker but natural notion of privacy opens the door to improvements in communication complexity over what is known for information theoretic privacy. In particular, assuming the existence of one-way functions, they present a two-server computational PIR scheme whose communication complexity is $O(n^\varepsilon)$, for every $\varepsilon > 0$.

Kushilevitz and Ostrovsky [20] observed that the linear lower bound on communication complexity (see Section 5.1) ceases to hold for computational privacy. Indeed, assuming the intractability of the Quadratic Residuosity problem, they presented a *single*-server computational PIR scheme whose communication complexity is $O(n^\varepsilon)$, for every $\varepsilon > 0$.

Gertner *et. al.* [17] have extended the privacy requirement so that the database's privacy is protected too. Specifically, they consider information retrieval protocols where the only information about the database contents revealed in one invocation is a single physical bit. They present general transformations of PIR schemes satisfying certain properties into PIR schemes that guarantee the database privacy as well, with a logarithmic overhead in the communication complexity. These transformations are applicable to all known k-server PIR schemes, for $k \geq 2$.

1.4 Organization

In Section 2 we introduce the model and discuss its attributes. In Section 3 we present the main results of this paper – a number of PIR schemes. Section 4 extends the problem to the case where we are interested in retrieving a block of bits (and not only a single bit). Section 5 provides some simple lower bounds for private retrieval schemes with a single server and for schemes with more servers but very restricted types of queries.

2 Model, Definitions, and Discussion

In this section we define a special case of private information retrieval schemes, where the interaction is carried out in one round. We also discuss the motivation underlying the definition and possible extensions thereof.

2.1 Definitions

Following the discussion in the introduction, we consider a randomized strategy for the user, which on input an index $i \in [n] \triangleq \{1, \ldots, n\}$ and random input r (of length ℓ_{rnd}), produces k queries (of length ℓ_{q} each), $Q_1(i, r), \ldots, Q_k(i, r)$, one per server. The servers respond according to strategies $A_1, \ldots, A_k$, with replies (of length ℓ_{a}) that depend on the contents of the database, denoted x, and the corresponding query. The user reconstructs the desired bit x_i from these k replies, together with i and r. The *privacy* requirement is that each individual query is distributed independently of i and thus the server gains no information about the identity of the desired item.

Definition 1 (Private Information Retrieval – one-round schemes): *A k-server* Private Information Retrieval (PIR) *scheme for database length n consists of*

- k query functions, $Q_1, \ldots, Q_k : [n] \times \{0,1\}^{\ell_{\mathrm{rnd}}} \mapsto \{0,1\}^{\ell_{\mathrm{q}}}$;
- k answer functions, $A_1, \ldots, A_k : \{0,1\}^n \times \{0,1\}^{\ell_{\mathrm{q}}} \mapsto \{0,1\}^{\ell_{\mathrm{a}}}$;
- a reconstruction function, $R : [n] \times \{0,1\}^{\ell_{\mathrm{rnd}}} \times (\{0,1\}^{\ell_{\mathrm{a}}})^k \mapsto \{0,1\}$.

These functions should satisfy

Correctness: *For every $x \in \{0,1\}^n$, $i \in [n]$, and $r \in \{0,1\}^{\ell_{\mathrm{rnd}}}$*

$$R(i, r, A_1(x, Q_1(i, r)), \ldots, A_k(x, Q_k(i, r))) \quad = \quad x_i \ .$$

Privacy: *For every $i, j \in [n]$, $s \in [k]$, and $q \in \{0,1\}^{\ell_{\mathrm{q}}}$,*

$$\mathbf{Pr}(Q_s(i, r) = q) \quad = \quad \mathbf{Pr}(Q_s(j, r) = q)$$

where the probabilities are taken over uniformly chosen $r \in \{0,1\}^{\ell_{\mathrm{rnd}}}$.

This definition can be extended to multi-round protocols, but since all our schemes are just one-round schemes, such an extension is not needed in the present work. Actually, in our schemes all servers have the same answer function (i.e., $A_1 = \ldots = A_k$), and the k query functions are very similar.

2.2 Discussion

We now spell out some of the attributes of the above definition.

Perfect Privacy: The privacy requirement stated as equality of the two probability distributions is an information theoretic notion. It means that even a (possibly malicious) computationally-unbounded server cannot get any indication on the identity of the desired item. Relaxations such as requiring statistical proximity or even just computational indistinguishability may suffice in certain contexts (cf., [10]).

Memoryless protocol: Both the user and server strategies are history independent. Thus, correctness and privacy are maintained even when different users (some of them possibly malicious) execute the protocols sequentially and/or concurrently.

Deterministic server strategies: Randomized server strategies do not offer any advantage over deterministic ones in our context (where the concerns are correctness and privacy of the *user*). However, randomized server strategies are essential for *database privacy* as considered in [17].

Non-collusion: We assume that the servers do not collude in trying to violate the user's privacy. This does not necessarily mean that they are forbidden from communicating. Such communication may be necessary for other reasons such as updating the contents of the database. We view the servers as providers of private access to the database. A detected violation of the privacy guarantees will result in severe damage to the server. It is as if a bank was caught in fraud. Thus, collusion is way too risky from the server's point of view. In the rare case where the user values its potential loss as more substantial than the server's risk, the user should not use a PIR scheme in which privacy depends on a non-collusion assumption. The single-server computational PIR scheme of [20] addresses this concern.

Coalitions: Information-theoretic PIR schemes which tolerate collusions of up to a certain number of servers provide an alternative answer to the above concerns. Specifically, the privacy requirement in Definition 1 can be generalized to coalitions of up to $t < k$ servers by requiring that, for every $s_1, ..., s_t \in [k]$, the joint distribution of $(Q_{s_1}(i, r), ..., Q_{s_t}(i, r))$, where r is uniformly distributed over $r \in \{0, 1\}^{\ell_{\mathrm{rnd}}}$, is independent of i. Such schemes have been presented in [12] (see Subsection 1.1).

2.3 Extensions

PIR of Blocks: In a more realistic model of private information retrieval, the data is organized in records (or blocks) rather than single bits. Clearly, a block may be retrieved by retrieving each of its bits, but significant saving is possible by taking advantage of the block structure (i.e., the fact that we only need to hide the identity of the block). Such schemes are presented in Section 4.

Search by Keywords: Throughout this work, we assume that the user knows the physical location of the information that it is interested in. A more realistic model allows the user to retrieve information based on keywords. Such schemes are presented in [11].

2.4 Complexity Measures

The main complexity measures we are interested in is the number of servers, k, and the communication complexity of the protocols, considered as a function of n and k. In addition, we require that all computations (of the user and the servers) be polynomial-time in n. Actually, in all our schemes, the server's action can be implemented in time *linear* in n, and the user's actions can be implemented in time linear in the communication complexity (which is typically sublinear in n).

In some applications, the (servers') computational overhead may be considered too high. In such cases, trade-offs between privacy and computational overhead may be considered. Specifically, the database may be partitioned (possibly at random) into several portions. The user wishing to retrieve an item will disclose the identity of the relevant portion, and invoke the PIR scheme to obtain the specific item from this portion. Clearly, the identity of the item within the portion will remain unknown to each individual server; yet the server's computation will now be linear in the *portion* length.

3 Single Bit PIR Schemes

All PIR schemes presented in this section are of the "linear summation" type. In these schemes, the user sends queries in the form of sequences of subsets $S_1, ..., S_t \subseteq \{1, \ldots, n\}$, and each server replies with a corresponding sequence of bits, $\oplus_{j \in S_1} x_j, ..., \oplus_{j \in S_t} x_j$. We start by describing a very simple 2-server scheme which is the basis of all the schemes in this section. In Subsection 3.2, we generalize the scheme to 2^d servers and in Subsection 3.3 we present an additional idea which allows to reduce the number of servers to about $2^d/d$ while maintaining about the same communication costs. Subsection 3.4 contains a generic transformation which may be applied to reduce the communication complexity of schemes in which the user's queries are longer than the servers' responses. This transformation is applicable to the schemes of Subsections 3.1 and 3.2, however the custom-made transformation in Subsection 3.3 yields superior results.

Notation: We use the following notations throughout the paper:

- $\mathcal{U}$ – a (generic) user.
- $\mathcal{SRV}_1, \ldots, \mathcal{SRV}_k$ – the servers.
- $x = x_1 \cdots x_n$ – a string in $\{0,1\}^n$, known to each server, representing the database.
- i – the index in x in which $\mathcal{U}$ is interested.
- $[m] \stackrel{\triangle}{=} \{1, 2, ..., m\}$.

For a set S and an element a, let

$$S \oplus a \stackrel{\triangle}{=} \begin{cases} S \cup \{a\} & \text{if } a \notin S \\ S \setminus \{a\} & \text{if } a \in S \end{cases}$$

3.1 A Basic Two-Server Scheme

We start by describing a very simple PIR scheme that allows $\mathcal{U}$ to privately obtain the bit x_i by receiving a single bit from each of two servers. The user uniformly selects a random set $S \subseteq [n]$ (i.e., each index $j \in [n]$ is selected with probability $1/2$). The user sends S to $\mathcal{SRV}_1$ and $S \oplus i$ to $\mathcal{SRV}_2$. Each server, upon receipt of the message $I \subseteq [n]$, replies with a single bit which is the exclusive-or of the bits with indices in I (i.e., $\mathcal{SRV}_1$ replies with $\oplus_{j \in S} x_j$ whereas $\mathcal{SRV}_2$ replies with $\oplus_{j \in S \oplus i} x_j$). The user exclusive-ors the answers it has received, thus retrieving the desired bit x_i. Clearly, none of the servers has obtained any information regarding which index was desired by the user (as each of the servers obtains a uniformly distributed subset of $[n]$).

Although the above scheme is less obvious than a solution in which one server sends all n bits to the user, it is not superior as far as the total amount of communication goes. Indeed each server sent only a single bit, but the messages sent by the user (specifying arbitrary subsets of $[n]$) are n bits long. Yet, this simple scheme serves as a basis for more efficient ones.

3.2 A Multi-Server Scheme

In this subsection, we present a scheme for any number $k \geq 2$ of servers. The scheme presented here, is combined with the covering codes method that we present in the next subsection to yield improved results.

The scheme presented in this subsection allows the user to obtain the desired bit by asking queries to $k = 2^d$ servers, for any $d \geq 1$, and requires total communication of $2^d \cdot (d \cdot n^{1/d} + 1)$. The key idea is to associate $[n]$ with the d-dimensional cube $[\ell]^d$ and generalize the simple scheme of Subsection 3.1, which may be viewed as the 1-dimensional case (i.e., $d = 1$). In the generalization, each of the 2^d servers is queried for the exclusive-or of the bits in a uniformly distributed (generalized) subcube. As in the basic scheme, the different subcubes are related, and this allows to retrieve the desired bit. The saving in communication comes from the fact that subcubes can be described more succinctly than general subsets.

We assume, without loss of generality that $n = \ell^d$. We embed x in a d-dimensional cube, associating each position $j \in [n]$ with a d-tuple $(j_1, ..., j_d) \in [\ell]^d$, in a natural manner. In particular, the index i of the desired bit is associated with a d-tuple $(i_1, \ldots, i_d) \in [\ell]^d$. It will also be convenient to associate the $k = 2^d$ servers with strings in $\{0,1\}^d$. The scheme works as follows.

1. $\mathcal{U}$ chooses uniformly and independently d random subsets $S_1^0, S_2^0, \ldots, S_d^0 \subseteq [\ell]$. Based on these subsets it defines another d subsets of $[\ell]$ by $S_1^1 = S_1^0 \oplus i_1, S_2^1 = S_2^0 \oplus i_2, \ldots, S_d^1 = S_d^0 \oplus i_d$. These $2d$ subsets are paired in a natural way; namely, $(S_1^0, S_1^1), \ldots, (S_d^0, S_d^1)$. To each of the $k = 2^d$ servers $\mathcal{U}$ sends a single subset per each pair, corresponding to the name of the server. Namely, for every $\alpha = \sigma_1 \cdots \sigma_d \in \{0,1\}^d$, the user sends the subsets $S_1^{\sigma_1}, S_2^{\sigma_2}, \ldots, S_d^{\sigma_d}$ to $\mathcal{SRV}_\alpha$.

2. Upon receiving the d subsets $S_1^{\sigma_1}, S_2^{\sigma_2}, \ldots, S_d^{\sigma_d}$, the corresponding server (i.e., $\mathcal{SRV}_{\sigma_1 \cdots \sigma_d}$) replies with the exclusive-or of the bits in the subcube defined by these subsets. Namely, $\mathcal{SRV}_{\sigma_1 \cdots \sigma_d}$ replies with the bit
$$\bigoplus_{j_1 \in S_1^{\sigma_1}, \ldots, j_d \in S_d^{\sigma_d}} x_{j_1, \ldots, j_d}.$$

3. The user exclusive-ors the $k = 2^d$ bits it has received.

The correctness of the above scheme can be proved in several ways. For example, this can be done by induction on d. Alternatively, one may consider the contribution of each bit $x_{j_1,\ldots,j_d}$ of the database to the sum computed by the user (in Step 3). This contribution depends on the number of subcubes (corresponding to the queries directed to the 2^d servers) which contain the position $(j_1, ..., j_d)$. It is not hard to see that $(i_1, ..., i_d)$ is the only position which is contained in an odd number of subcubes. Actually position $(i_1, ..., i_d)$ appears in a single subcube. This is because, for every $t \in [d]$, the value i_t appears in exactly one of the sets S_t^0, S_t^1. Each of the other positions $(j_1, ..., j_d)$ (i.e., those $\neq (i_1, ..., i_d)$) appears in an even number of subcubes: Suppose $j_t \neq i_t$, then for every $\sigma_1, ..., \sigma_d$,

$$\begin{array}{rcl} (j_1, ..., j_d) & \in & S_1^{\sigma_1} \times \cdots \times S_{t-1}^{\sigma_{t-1}} \times S_t^0 \times S_{t+1}^{\sigma_{t+1}} \times \cdots \times S_d^{\sigma_d} \\ & & \text{if and only if} \\ (j_1, ..., j_d) & \in & S_1^{\sigma_1} \times \cdots \times S_{t-1}^{\sigma_{t-1}} \times S_t^1 \times S_{t+1}^{\sigma_{t+1}} \times \cdots \times S_d^{\sigma_d} \quad . \end{array}$$

Therefore, in the sum modulo 2 computed by the user (in Step 3), the contribution of these positions is cancelled and the only value that remains is that of position $(i_1, ..., i_d)$.

The privacy of the above scheme follows by observing that not only each of the subsets $S_1^0, S_2^0, \ldots, S_d^0$ is a random subset of $[\ell]$ but also each of the subsets $S_1^1, S_2^1, \ldots, S_d^1$ (since each S_t^1 is obtained by flipping the membership of one element in the random set S_t^0). Therefore, from the point of view of each server, it receives a sequence of d uniformly and independently chosen subsets of $[\ell]$. Thus, the queries to each server are distributed in the same way, for each possible value of $i = (i_1, \ldots, i_d)$.

The communication involved in the above scheme consists of sending a sequence of d subsets in $[\ell]$ to each server, and receiving a single bit back. Hence the total communication complexity is $k \cdot (d \cdot \ell + 1) = 2^d \cdot (d \cdot \sqrt[d]{n} + 1)$. We note that the communication in the present scheme is not "balanced" – The user sends $d \cdot n^{1/d}$ bits to each server, and receives a single bit from each in response. Interestingly, the improvement in Section 3.3 results by balancing the communication (in a way specific to the above scheme). A generic balancing technique is presented in Section 3.4.

3.3 The Covering Codes Scheme

In this subsection we describe a method based on *covering codes*. This method (essentially) maintains the total communication complexity of the schemes described in the previous subsection but reduces the number of participating servers. It is especially useful when the number of servers (i.e., k) is small (e.g., $k = 2$ and $k = 4$).

We start with an example. For $d = 3$, the scheme of the previous subsection consists of a user and $2^d = 8$ servers whose names are associated with the binary strings of length $d = 3$. The user sends a subcube defined by the sets $(S_1^{\sigma_1}, S_2^{\sigma_2}, S_3^{\sigma_3})$ to $\mathcal{SRV}_{\sigma_1\sigma_2\sigma_3}$ which replies with the exclusive-or of the bits residing in this subcube. Thus, $3\sqrt[3]{n}$ bits are sent from the user to each server, which replies with a single bit. The key idea in the improvement is that $\mathcal{SRV}_{000}$, which gets the query (S_1^0, S_2^0, S_3^0), can produce a relatively short string which contains the answer to the query (S_1^0, S_2^0, S_3^1), sent to $\mathcal{SRV}_{001}$. Specifically, it knows S_1^0 and S_2^0 and it also knows that S_3^1 is of the form $S_3^0 \oplus j$, for some $j \in \{1, 2, ..., \sqrt[3]{n}\}$. Thus, $\mathcal{SRV}_{000}$ can emulate $\mathcal{SRV}_{001}$ by sending the $\sqrt[3]{n}$ bits corresponding to the $\sqrt[3]{n}$ possible queries which could have been sent to $\mathcal{SRV}_{001}$. In the same fashion, $\mathcal{SRV}_{000}$ can also emulate both $\mathcal{SRV}_{010}$ and $\mathcal{SRV}_{100}$. Thus, by letting $\mathcal{SRV}_{000}$ emulate $\mathcal{SRV}_{100}$, $\mathcal{SRV}_{010}$ and $\mathcal{SRV}_{001}$, and letting $\mathcal{SRV}_{111}$ emulate $\mathcal{SRV}_{011}$, $\mathcal{SRV}_{101}$ and $\mathcal{SRV}_{110}$, we get a scheme for two servers with total communication complexity $O(\sqrt[3]{n})$. We note that it is too expensive to let $\mathcal{SRV}_{000}$ emulate $\mathcal{SRV}_{011}$ as this will require considering all $(\sqrt[3]{n})^2$ possibilities for (S_2^1, S_3^1).

In general, the above "emulation" method depends on the ability to cover the strings in $\{0,1\}^d$ by few d-bit long string, where each string may cover itself and all strings at Hamming distance 1 from it. In other words, we consider the problem of covering $\{0,1\}^d$ by balls of radius 1 (in the Hamming geometry). This is a well known problem in coding theory. A **covering code**, C_d, *with radius* 1 *for* $\{0,1\}^d$ is a collection $C_d = \{c_1, c_2, \ldots, c_k\} \subseteq \{0,1\}^d$, such that the balls of radius 1 around the codewords cover the space; namely,

$$\{0,1\}^d \subseteq \cup_{c_j \in C_d} B(c_j, 1)$$

where $B(c,1)$ is the set of all d-bit long strings which differ from c in at most one position.

Given a (radius 1) covering code, $C_d = \{c_1, c_2, \ldots, c_k\}$ (for $\{0,1\}^d$), we use the emulation method to derive a k-database protocol of communication complexity $O(d \cdot k \cdot n^{1/d})$. The user, being interested in position $i = (i_1, \ldots, i_d)$, picks uniformly $S_1^0, S_2^0, \ldots, S_d^0 \subseteq [n^{1/d}]$, and sets $S_1^1 = S_1^0 \oplus i_1, S_2^1 = S_2^0 \oplus i_2, \ldots, S_d^1 = S_d^0 \oplus i_d$. The user sends to $\mathcal{SRV}_c$ ($c \in C_d$) the subcube corresponding to codeword c (i.e., $(S_1^{\sigma_1}, \ldots, S_d^{\sigma_d})$ where $c = \sigma_1 \cdots \sigma_d$). Each server $\mathcal{SRV}_c$ replies by emulating itself (i.e., one bit) and the servers corresponding to the words covered by the codeword c (i.e., $n^{1/d}$ bits per each such server). All these answers allow the user to compute the answer it would have received in the protocol for 2^d servers, and consequently retrieve the desired bit. The privacy of the original 2^d-server scheme is clearly preserved (since the queries to each $\mathcal{SRV}_c$ are chosen in the same way as in the 2^d-server scheme; it is only the answer function that is different). As for the communication complexity of the new protocol, we note that $d \cdot n^{1/d}$ bits are sent from $\mathcal{U}$ to each server and that the total number of bits sent back is $k + (2^d - k) \cdot n^{1/d}$ (note that only the emulation of servers corresponding to non-codewords requires $n^{1/d}$ bits and that it suffices to emulate/cover each such server once[1]). Thus, the total communication equals $(kd + 2^d - k) \cdot n^{1/d} + k$, and we get

Theorem 1: Let d and k be integers so that there is a k-word covering code (of radius 1) for $\{0,1\}^d$. Then there exists a private information retrieval schemes for k servers, each holding n bits of data, so that the communication complexity of the scheme is $k + (2^d + (d-1) \cdot k) \cdot n^{1/d}$.

Clearly, k in the above theorem need not be greater than 2^d. On the other hand, since every radius 1 ball contains exactly $d+1$ points in $\{0,1\}^d$, the number of codewords k satisfies $k \geq \frac{2^d}{d+1}$ (this is the *volume bound*, cf., [16]). This lower bound is not always attainable. The construction given above, for $d = 3$, uses the fact that $\{(0,0,0),(1,1,1)\}$ is a covering code with radius 1 of $\{0,1\}^3$. For $d = 4$ there exist covering codes with four codewords (e.g., $\{(0,0,0,0),(1,1,1,1),(1,0,0,0),(0,1,1,1)\}$) but not with fewer codewords (due to the volume bound). In Figure 1 we list the best known covering codes for d up to 8, the corresponding volume bounds, and the communication complexity of the resulting protocol (i.e., $(2^d + (d-1)k) \cdot n^{1/d}$, ignoring the additive term of k). We note that all these covering codes are optimal (minimum size) [19]. For $d = 3$ and $d = 7$, these are Hamming Codes which are perfect codes (all balls are disjoint).

As one can see from this table, the improvement derived by the emulation method (over the simpler method of Section 3.2 which requires 2^d servers) is quite meaningful for small values of d. Covering codes with larger radii (say 2 or 3) are also applicable in principle. For example, a k word

[1] Formally, we consider a fixed *exact* cover of $\{0,1\}^d$ by sets $S(c_j)$'s so that $S(c_j) \subseteq B(c_j, 1)$, for every $j = 1, \ldots, k$.

dimension (i.e., d)	2^d	# codewords (# servers) (i.e., k)	volume (lower) bound	total communication asymptotic	$n = 2^{20}$	$n = 2^{30}$	$n = 2^{40}$
3	8	2	2	$12n^{1/3}$	1,224	12,300	123,864
4	16	4	4	$28n^{1/4}$	924	5,096	28,700
5	32	7	6	$60n^{1/5}$	1,020	3,900	15,420
6	64	12	10	$124n^{1/6}$	1,249	3,968	12,598
7	128	16	16	$224n^{1/7}$	1,792	4,480	11,872
8	256	32	29	$480n^{1/8}$	2,715	6,458	15,360

Figure 1: Covering Codes and PIR complexity

radius 2 covering code of $\{0,1\}^d$ would yield communication complexity $k \cdot d \cdot n^{1/d} + k \cdot \binom{d}{2} \cdot n^{2/d}$. Reviewing the parameters of the best codes [19], they turn out to be inferior for our purposes than the radius 1 codes.

The results using the covering codes methods are most appealing for the cases of 2 and 4 servers. These cases are summarized in the next corollary to Theorem 1.

Corollary 2: There are private information retrieval schemes for n bits data, with the following parameters:

- For two servers (i.e., $k = 2$), the communication complexity is $12\sqrt[3]{n} + 2$.
- For four servers (i.e., $k = 4$), the communication complexity is $28\sqrt[4]{n} + 4$.

As noted above, for d dimensional space the communication complexity of these schemes is $(2^d + (d-1)k) \cdot n^{1/d}$. As $k \geq 2^d/(d+1)$, this is $\Omega(k \log k n^{1/(\log k + \log\log k)})$.

Remark: Note that the user's computation in the above PIR schemes can be done in time linear in the communication complexity. In addition, for each of the servers the computation time is only linear in n. To see this, note that $\mathcal{SRV}_c$ (for $c \in C_d$) needs to compute the exclusive-or of the bits residing in the subcube corresponding to codeword c. In addition, for each of the words c' in Hamming distance 1 from c, the server $\mathcal{SRV}_c$ needs to compute the exclusive-or of the bits in $n^{1/d}$ subcubes. However, by the structure of these subcubes each such exclusive-or can be obtained from the bit computed for the subcube corresponding to c and the examination of $n^{(d-1)/d}$ bits. All together $(d+1) \cdot n$ bits are xored during the server's computation.

3.4 A Generic Transformation

Consider an arbitrary PIR scheme in which the communication is carried out in one round (i.e., the user simultaneously queries each server and receives answers from which it computes the desired bit). Given such a scheme for databases containing n bits, one can derive a scheme for databases containing $m \cdot n$ bits by repeating the scheme in parallel as follows. The user views the $m \cdot n$ bits as a m-by-n matrix of bits. To retrieve the (j,i)-th bit in the matrix, $\mathcal{U}$ executes the n-bit scheme with i being the desired bit (ignoring, for the time being, the value of j). That is, the user sends the same query as it would have send in the n-bit scheme when being interested in the i^{th} bit. Now, each server sends m responses to the single query it has received. These answers correspond to m different executions of the n-bit scheme, each one with a different row (an n-bit string). Specifically, in the j-th execution ($j = 1, ..., m$), the server computes its response with respect to the j-th row. Thus, the user privately retrieves the entire i-th column of the matrix, from which it finds the desired (j, i)-th bit. Let us compare the communication complexity of the original n bits scheme with the resulting $m \cdot n$ bits scheme. The communication from the user to each server remains unchanged, while the communication in the server-to-user direction increases by a factor of m.

In case the original PIR is such that the user sends longer messages than it receives, the above transformation allows to reduce the total communication complexity. Specifically, applying the above transformation to the Basic Scheme (of Subsection 3.1), we obtain a 2-server PIR with communication complexity $2 \cdot (n + m)$ for a database of $m \cdot n$ bits. Thus,

Corollary 3: There is a 2-server private information retrieval scheme, for n bits data, with communication complexity $4\sqrt{n}$.

We comment that the above transformation has been applied in [12, Sec. 4] to improve the communication complexity of the k-server PIR schemes (based on polynomial interpolation and omitted from this version) from $O(\sqrt[k-1]{n})$ to $O(\sqrt[k]{n})$.

4 Private Block Retrieval

In this section we consider a more realistic model of private information retrieval in which the data is partitioned into blocks (or records) rather than single bits. For simplicity, we assume that each block/record contains the same number of bits, ℓ. We denote by $\mathcal{PIR}_k(n, \ell)$ the problem of retrieving privately an (ℓ-bit long) information block from k servers, each holding the same n blocks (notice that the overall contents is $n \cdot \ell$ bits). Previous sections have dealt with $\mathcal{PIR}_k(n, 1)$. Clearly $\mathcal{PIR}_k(n, \ell)$ can be solved by ℓ invocations of $\mathcal{PIR}_k(n, 1)$ (i.e., by considering in the j-th invocation only the j-th bit of each of the n blocks), but there are much more efficient reductions of $\mathcal{PIR}_k(\cdot, \ell)$ to $\mathcal{PIR}_k(\cdot, 1)$.

4.1 Transformations

We start by noting that the generic transformation of Subsection 3.4 actually provides such a reduction. Specifically,

Proposition 4: Suppose that $\mathcal{PIR}_k(n, 1)$ can be solved by a one-round protocol in which the user sends $\alpha_k(n)$ bits to each server and receives $\beta_k(n)$ bits in return (from each server). Then, for every $\ell > 1$, $\mathcal{PIR}_k(n, \ell)$ can be solved by a one-round protocol in which the user sends $\alpha_k(n)$ bits to each server and receives $\ell \cdot \beta_k(n)$ bits in return (from each server).

In Section 3.4 we emphasized the asymmetric effect that the above transformation has on the communication complexity – increasing the communication from the servers to the user while maintaining the communication complexity in the other direction. We now present an "asymmetric" transformation in the opposite direction.

Proposition 5: Suppose that $\mathcal{PIR}_k(n, 1)$ can be solved by a one-round protocol in which the user sends $\alpha_k(n)$ bits to each server and receives one bit in return (from each server). Furthermore, suppose that the user reconstructs the desired information bit by computing $\sum_{p=1}^{k} \tau_p$ (modulo 2), where τ_p is the message obtained from $\mathcal{SRV}_p$. Then, for every $m > 1$, $\mathcal{PIR}_k(m \cdot (n-1), 1)$ can be solved by a one-round protocol in which the user sends $m \cdot \alpha_k(n)$ bits to each server and receives one bit in return (from each server).

We note that the schemes presented in Subsections 3.1–3.2 (but not those derived in Subsections 3.3–3.4) meet the hypothesis of the proposition. Furthermore, the proposition can be generalized to $\mathcal{PIR}_k(\cdot, \ell)$ schemes (in which each bit in the block is computed as conditioned above).

Proof: Partition the $N \stackrel{\triangle}{=} m \cdot (n-1)$ bits in the database, into m strings. Each string holds $n-1$ original bits and is augmented by a dummy bit at the n^{th} position that is set to zero. Bit positions in $[N]$ are represented as pairs in $[m] \times [n-1]$ in a natural manner. The user, wishing to retrieve $i = [i_1, i_2] \in [m] \times [n-1]$, employs the $\mathcal{PIR}_k(n, 1)$ scheme in parallel m times. In the j^{th} instance $\mathcal{U}$ behaves as when asking for position i_2 if $j = i_1$, and as asking for position n (the dummy bit) otherwise. Each server adds together (modulo 2) the answers it would have sent in each of the m invocations of $\mathcal{PIR}_k(n, 1)$ and sends this sum as its only message. The user just adds all answers it has obtained and this is the retrieved bit. We emphasize that each server sends only a single bit rather than m such bits. The new scheme clearly satisfies the privacy requirement. Correctness follows from associativity of addition, and the fact that the dummy position (i.e., position n) is set to 0. That is, let τ_p^j be the designated answer of $\mathcal{SRV}_p$ in the j^{th} invocation. The answer bit sent by $\mathcal{SRV}_p$ in the above scheme is just $\sum_{j=1}^{m} \tau_p^j$. We also know (by the correctness of the $\mathcal{PIR}_k(n, 1)$ scheme) that $\sum_p \tau_p^j$ equals x_i if $j = i_1$, and 0 otherwise. Thus,

$$\sum_{p=1}^{k} \sum_{j=1}^{m} \tau_p^j = \sum_{j=1}^{m} \sum_{p=1}^{k} \tau_p^j = x_i \quad ,$$

as needed. $\square$

The requirements from the schemes in Proposition 5 seem quite restrictive. One may want to consider the following more general scenario. Suppose that $\mathcal{PIR}_k(n, 1)$ can be solved by a one-round protocol in which the user sends $\alpha_k(n)$ bits to each server and receives $\beta_k(n)$ bits in return (from each server). Furthermore, suppose that the user reconstructs the desired information bit by computing $g(\sum_{p=1}^{k} f_p(\gamma_p))$, where γ_p is the message obtained from $\mathcal{SRV}_p$, the f_p's are arbitrary fixed functions mapping binary strings into elements of some finite Abelian group (of cardinality at most $2^{\beta_k(n)}$), summation is done over this group and g is a homomorphism of the group onto Z_2. Then we claim that, for every $m > 1$, $\mathcal{PIR}_k(m \cdot (n-1), 1)$ can be solved by a one-round protocol in which the user sends $m \cdot \alpha_k(n)$ bits to each server and receives a single bit in return (from each server). (We stress that both g and the f_p's may not depend on the desired bit nor on the randomness used by $\mathcal{U}$.) To see this, simply observe that any such $\mathcal{PIR}_k(n, 1)$ scheme can be transformed into a $\mathcal{PIR}_k(n, 1)$ scheme of the form required by Proposition 5. This is done by letting each server compute the value of $g(f_p(\gamma_p))$ (a single bit) and sending it to the user. Clearly, this does not violate the privacy of the scheme. Moreover, since g is a homomorphism, it follows that

$$\sum_{p=1}^{k} g(f_p(\gamma_p)) = g\left(\sum_{p=1}^{k} f_p(\gamma_p^j)\right) \quad ,$$

and so by computing the sum of the answers the user retrieves the desired bit.

Combining the above two propositions, we obtain:

Corollary 6: Let $\mathcal{PIR}_k(n,1)$ be as in Proposition 5 and $\ell, m > 1$. Then, $\mathcal{PIR}_k(m\cdot(n-1),\ell)$ can be solved by a one-round protocol in which the user sends $m\cdot\alpha_k(n)$ bits to each server and receives ℓ bits in return (from each server). In particular, $\mathcal{PIR}_k(n,\ell)$ can be solved within ℓ times the complexity of $\mathcal{PIR}_k(\frac{n}{\ell}+1,1)$.

The above should be contrasted with ℓ times the complexity of $\mathcal{PIR}_k(n,1)$, obtained in the straightforward manner.

4.2 Corollaries

In some settings the number of records is not substantially bigger than the length of individual records. In these settings the overhead introduced by private information retrieval is quite small, compared to non-private information retrieval. We exemplify two such cases – one with $n \leq \ell$, the other with $n \leq \ell^2/4$. We exhibit simple linear schemes for these two cases, with *constant* multiplicative overhead, using $k=2$ and $k=4$ servers, respectively. The first example, with $n \leq \ell$, employs the basic two-servers scheme (of Section 3.1), and the total communication overhead is just a factor of 4.

Corollary 7: Let $n \leq \ell$, then $\mathcal{PIR}_2(n,\ell)$ can be solved by a one-round protocol of total communication complexity $4\cdot\ell$.

The above is to be compared to $\ell + \log n$ bits required in "non-private" retrieval of an ℓ-bit long block (from a database holding n such blocks).

Proof: We use the $\mathcal{PIR}_2(n,1)$ scheme (of Section 3.1) in which $\mathcal{U}$ sends $\alpha_2(n) = n$ bits to each server (indicating a subset of the bits in the database), and receives $\beta_2(n) = 1$ bit from each (the exclusive-or of these bits). Using Proposition 4, we get a $\mathcal{PIR}_2(n,\ell)$ scheme with total communication $2(\alpha_2(n) + \ell\beta_2(n)) = 2n + 2\ell \leq 4\ell$ □

Corollary 8: Let $n \leq \ell^2/4$, then $\mathcal{PIR}_4(n,\ell)$ can be solved by a one-round protocol of total communication complexity $8\cdot\ell$.

Proof: We use the $\mathcal{PIR}_4(n,1)$ scheme (of Section 3.2, $d=2$) in which $\mathcal{U}$ sends $\alpha_4(n) = 2\sqrt{n}$ bits to each server (indicating a "two dimensional subcube" of the bits in the database), and receives $\beta_4(n) = 1$ bit from each (the exclusive-or of these bits). Using Proposition 4, we get a $\mathcal{PIR}_4(n,\ell)$ scheme with total communication $4(\alpha_4(n) + \ell\beta_4(n)) = 8\sqrt{n} + 4\ell \leq 8\ell$. □

Of course larger values of d may be used to yield constant overhead schemes with $n = O(\ell^d)$ and $k = 2^d$ servers. However, we believe the two schemes presented above are the ones of interest for realistic size databases. For example, the two server scheme is applicable to records of sizes 2^{15} and 2^{20} for databases containing 2^{30} and 2^{40} bits, respectively. The four server scheme is applicable to records of sizes 2^{10} and 2^{13} for databases containing 2^{30} and 2^{40} bits, respectively.

Note that unlike the $\mathcal{PIR}_2(n,1)$ scheme (of Section 3.1), the obvious $\mathcal{PIR}_2(n,1)$ (or actually $\mathcal{PIR}_1(n,1)$) in which each database sends its contents to the user who then reconstructs the desired bit, does not satisfy the hypothesis of Proposition 4 (since the reconstruction depends on i).

5 Lower Bounds

The question of proving lower bounds on private information retrieval schemes remains one of the most intriguing open problems of this paper. The only obvious lower bound is $\log n$ bits which holds for any number of servers (this follows from communication complexity considerations without using any privacy argument). In Subsection 5.2 we prove lower bounds for schemes of very restricted form. We start, however, with a simple lower bound on the communication complexity in SINGLE-server (information theoretic) PIR schemes.

5.1 The Single Server Case

We prove that if there is only one copy of the database available then n bits must be exchanged and hence the trivial solution is optimal in this case. The lower bound holds even if the communication between the user and the database allows interaction (i.e., not only a single query and an answer to it). The bound is clearly due to the (information-theoretic) privacy constraint; otherwise, $\log_2 n + 1$ bits are enough ($\mathcal{U}$ sends i and gets back x_i).

We say that a communication C is *possible* for (x, i) if when the database content is x and the user is interested in the i-th bit there is a positive probability for C to be the communication. We say that a communication C is *possible* for i if it is possible for some pair (x, i). Now, fix a value i and assume towards a contradiction that the number of possible communications for i is smaller than 2^n. This implies that there exist $x \neq y$ and C such that C is possible for both (x, i) and (y, i) (otherwise, if the sets of possible communications for (x, i) on the various x's are disjoint then there must be at least 2^n such communications). By the privacy requirement, for every $j \in [n]$, C must also be possible for (x, j), (or else $\mathcal{SRV}$ distinguishes the item identities i from j on database contents x). Similarly, C is possible for (y, j), for every $j \in [n]$. Thus, in particular, C is possible for both (x, j) and (y, j), where j is an index for which $x_j \neq y_j$. This yields contradiction since on the same communication, C, the user must output the same bit, and so this output cannot equal both x_j and y_j.

5.2 Linear Summation Queries with a single-bit Answers

In an attempt to develop lower bounds for the problem, we consider the very simple case in which there are two servers and the user makes a single *linear summation* query to each of them. In this simple case we were able to show that privacy requires the user to send long messages (i.e., of length linear in the length of the database). This lower bound is very restricted with respect to what we want, but on the other hand it provides yet another demonstration of the strength of the privacy condition.

We consider the case of $k = 2$ servers. We restrict our attention to schemes in which each of the two servers is asked a query and answers with a single bit. Moreover, we insist that the scheme is of the "linear summation" type. That is, each query is just a name of a vector (set) q and the answer is $\oplus_{i:q_i=1} x_i$. The user takes the two bits b_1, b_2 received from $\mathcal{SRV}_1, \mathcal{SRV}_2$ (respectively) and computes $b_1 \oplus b_2$. Recall that in Section 3.1 we proved the existence of such a scheme in which each of the queries sent by the user is n-bit long. We now show that this is essentially optimal.

We say that a query, q, is *possible* for $(i, 1)$ (resp., $(i, 2)$) if on input i there exists some sequence of coins which makes $\mathcal{U}$ send query q to $\mathcal{SRV}_1$ (resp., $\mathcal{SRV}_2$).

Claim: Suppose that q is possible for $(i,1)$. Then each query at even (resp., odd) Hamming distance from q is possible for $(i,1)$ (resp., $(i,2)$).

This claim implies that the set of possible queries for each server has cardinality at least 2^{n-1}, requiring a query description length of at least $n-1$ bits, which establishes the lower bound.

Proof: The proof is based on two observations.

1. By the privacy of the scheme, we have that for every $j,h \in [n]$, if the vector v is possible for $(j,1)$ then the vector v is also possible for $(h,1)$. (Otherwise, $\mathcal{SRV}_1$ distinguishes the item identities j from h.)

2. Let e_h denote the h^{th} unit vector (i.e., $e_h = 0^{h-1}10^{n-h}$). Then, by definition of the summation-type scheme, if, on input h, the user $\mathcal{U}$ makes query v to $\mathcal{SRV}_1$ and query w to $\mathcal{SRV}_2$ then it holds that $v \oplus w = e_h$. (Otherwise, we cannot have $x_h = (\oplus_{j':v_{j'}=1} x_{j'}) \oplus (\oplus_{j':w_{j'}=1} x_{j'})$.)

Thus, if v is possible for $(j,1)$ (resp., $(j,2)$) then, for every $h \in [n]$, the vector v is possible for $(h,1)$ (resp., $(h,2)$) and therefore $v \oplus e_h$ is possible for $(h,2)$ (resp., $(h,1)$) and also for $(j,2)$ (resp., $(j,1)$). Thus, each query at Hamming distance 1 from v is possible for $(j,2)$ (resp., $(j,1)$). The claim follows. $\square$

We note that, in fact, the upper bound of Section 3.1 can be improved so that $n-1$ bits (instead of n) are sent to each server. This is done by choosing a random subset S of even cardinality with uniform distribution among these subsets. Send S to $\mathcal{SRV}_1$, and $S \oplus i$ to $\mathcal{SRV}_2$. The subset $S \oplus i$ is uniformly distributed among odd cardinality subsets. To specify even (or odd) sets, $n-1$ bits suffice.

Acknowledgment

We wish to thank Muli Safra, Shafi Goldwasser, Don Coppersmith and Joe Kilian for helpful discussions regarding related issues, and Amos Beimel and Ehud Hausman for their comments on earlier drafts of this paper. We also thank Tuvi Etzion for pointers to the coding theory literature, and to Oded Shmueli for pointers to the database literature.

References

[1] Abadi M., J. Feigenbaum, and J. Kilian. On Hiding Information from an Oracle. *JCSS*, 39:1, pp. 21–50, 1989.

[2] N. Adam, and J. Wortmann. Security Control Methods for Statistical Databases: A Comparative Study. *ACM Computing Surveys*, 21:4, pp. 515–555, 1989.

[3] A. Ambainis. An Upper Bound on the Communication Complexity of Private Information Retrieval. *Proc. of 24th ICALP*, Springer, Lecture Notes in Computer Science, Vol. 1256, pages 401–407, 1997.

[4] L. Babai, P. Kimmel, and S. V. Lokam. Simultaneous Messages vs. Communication. *STACS*, pp. 361–372, 1995.

[5] D. Beaver and J. Feigenbaum. Hiding Instances in Multioracle Queries. *STACS*, Springer, Lecture Notes in Computer Science, Vol. 415, pp. 37-48, 1990.

[6] D. Beaver, J. Feigenbaum, J. Kilian and P. Rogaway. Security with Low Communication Overhead. *CRYPTO'90*, Springer, Lecture Notes in Computer Science, Vol. 537, pp. 62-76, 1991. This is an early version of [7].

[7] D. Beaver, J. Feigenbaum, J. Kilian and P. Rogaway. Locally Random Reductions: Improvements and Applications. *Journal of Cryptology*, 10:1, pp. 17-36, 1997.

[8] S. Ceri and G. Pelagatti. Distributed Database Principles & Systems. McGraw Hill, 1984.

[9] F. Chin. Security Problems on Inference Control for SUM, MAX, and MIN Queries. *JACM*, 33:3, pp. 451–464, 1986.

[10] B. Chor and N. Gilboa. Computationally Private Information Retrieval. In *29th STOC*, pp. 304–313, 1997.

[11] B. Chor, N. Gilboa, and M. Naor. Private Information Retrieval by Keywords. TR CS0917, Department of Computer Science, Technion, 1997.

[12] B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan. Private Information Retrieval. In *36th FOCS*, pp. 41-50, 1995.

[13] D. Denning. Cryptography and Data Security. Addison-Wesley, 1982.

[14] D. Dobkin, A. K. Jones, and R. J. Lipton. Secure Databases: Protection Against User Influence. *ACM Transactions on Database Systems*, 4:1, pp. 97–106, 1979.

[15] L. Fortnow and M. Szegedy On the Power of Two-Local Random Reductions. *Information Processing Letters*, 44:6, pp. 303–306, 1992.

[16] R.G. Gallager. Information Theory and Reliable Communication. John-Wiley and Sons, New-York, 1968.

[17] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. Manuscript, May 1997.

[18] O. Goldreich and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *JACM*, Vol. 43, No. 3, 1996, pp. 431–473.

[19] I.S. Honkala. Modified Bounds for Covering Codes. *IEEE Transactions on Information Theory*, 37:2, pp. 351–365, 1991.

[20] E. Kushilevitz and R. Ostrovsky. Replication is not Needed: Single Database, Computationally-Private Information Retrieval. In *38th FOCS*, pp. 364–373, 1997.

[21] R. Ostrovsky, V. Shoup. Private Information Storage. In *29th STOC*, pp. 294–303, 1997.

[22] P. Pudlák, and V. Rödl. Modified Ranks of Tensors and the Size of Circuits. *STOC*, pp. 523-531, 1993. This is an early version of [23].

[23] P. Pudlák, V. Rödl, and J. Sgall. Boolean Circuits, Tensor Ranks, and Communication Complexity. *SICOMP*, Vol. 26, pp. 605-633, 1997.

[24] R.L. Rivest. Private communication, quoted in the Acknowledgment Section of Abadi et al. [1].

[25] R.L. Rivest, L. Adleman, and M.L. Dertouzos. On Data Banks and Privacy Homomorphisms, *Foundations of Secure Computation* (eds., R. DeMillo, D. Dobkin, A. Jones, and R. Lipton). Academic Press, 1978.

[26] P. Tendick, and N. Matloff. A Modified Random Perturbation Method for Database Security. *ACM Transactions on Database Systems*, 19:1, pp. 47–63, 1994.

[27] J. D. Ullman. *Principles of Database Systems.* Second edition, 1982.