Private Information Retrieval

Benny Chor[*] Oded Goldreich[†] Eyal Kushilevitz[‡] Madhu Sudan[§]

ABSTRACT: 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 database gets no information on the identity of the item retrieved by the user.

For a single database, achieving this type of privacy requires communicating the whole database, or n bits (where n is the number of bits in the database). Our schemes use the replication to gain substantial saving. In particular, we have

- A two database scheme with communication complexity of $O(n^{1/3})$.
- A scheme for a constant number, k, of databases with communication complexity $O(n^{1/k})$.
- A scheme for $\frac{1}{3}\log_2 n$ databases with polylogarithmic (in n) communication complexity.

1 Introduction

Consider a user that makes a query in a database. A lot of research was devoted to methods that protect the database against a "curious" user. For example, methods that do not allow a user to ask queries to a *statistical database* in a way that enables him to reconstruct the value of particular entities (e.g., [2, 9, 10, 11, 18] and [19, 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 that if the user wants to keep its privacy then essentially the only thing he can do is to ask for a copy of the *whole database*; clearly, this is an undesirable solution.

The rapid development of distributed databases (see [8]) and all kind of data-services ("information highways") results in many scenarios in which the same database is replicated in several sites. This raises hope to get around the difficulty of achieving privacy in the single database scenario; it may be possible to make queries to several databases such that from the answers the desired information can be obtained, while from each query no information is given as to the information the user is interested in, hence its privacy is maintained.

Before going any further let us make the problem more concrete. We view the database as a string x of length n. Identical copies of this string are stored in $k \geq 2$ sites. The user has some index i, and he is interested in obtaining the value of the bit x_i.

We present various schemes that solve the retrieval problem 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-database scheme with communication complexity of $O(n^{1/3})$, a k-database scheme with complexity $O(n^{1/k})$ (for constant k), and a $(\frac{1}{3}\log_2 n+1)$-databases scheme with total communication complexity $\frac{1}{3}(1+o(1))\cdot\log_2^2 n\cdot\log_2\log_2(2n)$. We remark that our schemes can be modified (with a small penalty in the communication complexity) so as to guarantee a higher degree of privacy: for $t \leq k-1$, knowing t of the queries still gives no information as to what is the value of i that the user is interested in. We also remark that some of 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, 11, 19]).

Related Work

For the case case of $k = 2$ (i.e., two databases), a first indication that something better than the user asking for a copy of x can be done is given by a recent result of Pudlák and Rödl [16]. With a complexity-theory motivation in mind they studied the following question. There are three players: D_1 that holds a string x and an index j, D_2 that holds the same string x and an index ℓ, and U that knows both j and ℓ. The goal is for D_1 and D_2 to send a single message each to U so that he will be able to compute the bit $x_{j+\ell \bmod n}$. They show that this can be done using $o(n)$ bits (more precisely, $O(n\log_2\log_2 n/\log_2 n)$). Using their protocol, a scheme which guarantees privacy to the user of a database can be constructed as follows: The user chooses uniformly at random a pair of indices j,ℓ such that $j+\ell = i \mod n$. He sends j to the first database, ℓ to the second and the three of them execute the protocol. This solves the problem with $o(n)$ bits while maintaining privacy.

Independently of our work, Babai, Kimmel, and Lokam [5] studied a problem related to the one studied in [16] (where, again, the motivation comes from complexity theory). When viewed in our context their work provides a private information retrieval scheme for the case of general k with total communication $O(kn^{H_2(1/(k+1))})$. In particular for the case of $k = 2$ this provides a scheme with $O(n^{H_2(1/3)}) \approx O(n^{.92})$ communication. This is better than the scheme that can be derived from the work of [16] but still not as good as the scheme provided in this

[*] C.S. Dept., Technion, Haifa, Israel. benny@cs.technion.ac.il
[†] C.S. and Appl. Math. Dept., Weizmann Institute of Science, Rehovot, Israel. Supported by grant No. 92-00226 from the Israel–US Binational Science Foundation (BSF), Jerusalem, Israel. oded@wisdom.weizmann.ac.il
[‡] C.S. Dept., Technion, Haifa, Israel. eyalk@cs.technion.ac.il
[§] IBM T.J. Watson Research Center, U.S.A. madhu@watson.ibm.com

paper.

In [17, 1, 6, 7] the *instance hiding* problem is introduced and studied. In this problem, a computationally bounded player U that holds an 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 instance i hidden from the oracles. In a sense, this problem can be viewed as if the oracles have a string $f(1)f(2)\dots f(n)$ 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 [6, 7] are relevant to our problem, especially the use of low degree polynomials, introduced by Beaver and Feigenbaum [6], and further developed by Beaver, Feigenbaum, Kilian and Rogaway [7]. In particular, the scheme of [6] for $1 + \log_2 n$ databases is essentially the one we use as our starting point in Subsection 4.1. From the construction in [7] it is possible to derive a private information retrieval scheme for k (constant) databases with $O(n^{1/(k-1)})$ communication (see Remark 5.2 in [7]). In fact the scheme shown here in Subsection 4.2 can be considered an improved variant of their construction.

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 databases is poly-logarithmic in n. In contrast, the main thrust of our work is the case with small number of databases (specifically, smaller than $\log_2 n$). 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).

Organization

In Section 2 we introduce notations and basic definitions. In Section 3 we develop several schemes where every reply is an exclusive-or of a subset of the database's bits. In Section 4 we introduce methods based on low degree polynomial interpolation. Section 5 contains numeric results on the communication complexity for relevant numbers and sizes of databases. Section 7 describes a generalization, which guarantees privacy against coalitions of more than a single database. Conclusions and open problems can be found in Section 8.

2 Preliminaries and Definitions

Throughout the paper we denote the user by $\mathcal{U}$, the k databases by $\mathcal{DB}_1, \dots, \mathcal{DB}_k$, the identical data they hold by $x \in \{0,1\}^n$, and the bit in which $\mathcal{U}$ is interested by $i \in [m] \triangleq \{1, 2, ..., m\}$.

PRIVACY: We want that for every database $\mathcal{DB}_\ell$, for every possible content of the database, x, and any two indices i and j the database will not be able to distinguish between the case that the user holds index i and the case that the user holds index j. That is, the communication between the user and $\mathcal{DB}_\ell$ should be equally distributed, regardless of the index i. NOTATION FOR SECTION 3: For a set S and an element a let

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

NOTATION FOR SECTION 4: We use finite fields, denoted $GF(q)$, where q is a prime power. For notational simplicity, we denote the q elements of $GF(q)$ by $0, 1, ..., q-1$. Whenever adding or multiplying field elements it is understood that these operations are the field's operations.

3 The Linear-Summation Scheme

In this section we describe various schemes that are of the "linear summation" type. In these schemes, the user sends queries in the form of subsets $S \subseteq \{1, \dots, n\}$, and the database replies with $\oplus_{j \in S} x_j$.

3.1 A Basic Two-Databases Scheme

We start by describing a very simple scheme that allows the user $\mathcal{U}$ to privately obtain the bit x_i by receiving a single bit from each of two databases. 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 $\mathcal{DB}_1$ and $S \oplus i$ to $\mathcal{DB}_2$. Each of these databases, when receiving the message $I \subseteq [n]$, replies with a single bit which is the exclusive-or of the bits with indices in I (i.e., $\mathcal{DB}_1$ replies with $\oplus_{j \in S} x_j$ whereas $\mathcal{DB}_2$ 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 databases has obtained any information regarding which index was desired by the user (as each of the databases obtains a uniformly distributed subset of $[n]$).

Although the above scheme is less obvious than a solution in which one database sends all n bits to the user, it is not superior as far as the total amount of communication goes. Indeed each database 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-Database Scheme

In this subsection, we present a scheme for any number $k \geq 2$ of databases available. Later, in Section 4, we present a better scheme for the case of large k (i.e., $k > 4$). However, the scheme presented here, together with the covering codes method that we present in the next subsection leads to the best upper bounds we have for small values of k (the number of databases) and in particular for the interesting case $k = 2$.

The scheme presented in this subsection allows the user to obtain the desired bit by asking queries to $k = 2^d$ databases, 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 databases is queried for the exclusive-or of the bits in a uniformly distributed 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 the 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 $k = 2^d$ databases 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 the natural way; namely, $(S_1^0, S_1^1), \ldots, (S_d^0, S_d^1)$. To each of the $k = 2^d$ databases $\mathcal{U}$ sends one subset per pair, corresponding to the name of the database. 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 $\mathcal{DB}_\alpha$.
2. Upon receiving the d subsets $S_1^{\sigma_1}, S_2^{\sigma_2}, \ldots, S_d^{\sigma_d}$, the database (i.e., $\mathcal{DB}_{\sigma_1 \cdots \sigma_d}$) replies with the exclusive-or of the subcube defined by these subsets. Namely, $\mathcal{DB}_{\sigma_1 \cdots \sigma_d}$ 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 easily verified. The privacy of the above scheme follows by observing that each database receives a sequence of d uniformly and independently chosen subsets of $[\ell]$. Thus, the queries to each database 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 database, and receiving a single bit back. Hence the total communication complexity is $k \cdot (d \cdot \ell + 1) = 2^d \cdot (1 + d \cdot \sqrt[d]{n})$. We note that the communication in the present scheme is not balanced. The user sends $d \cdot n^{1/d}$ bits to each database, 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 4.3.

3.3 The Covering Codes Scheme

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

We start with an example. For $d = 3$, the scheme of the previous subsection consists of a user and $2^d = 8$ databases 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 $\mathcal{DB}_{\sigma_1\sigma_2\sigma_3}$ 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 database, which replies with a single bit. The key idea in the improvement is that $\mathcal{DB}_{000}$, 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 $\mathcal{DB}_{001}$. Specifically, it knows S_1^0 and S_2^0 and it also knows that S_3^1 must be one of form $S_3^0 \oplus j$, for some $j \in \{1, 2, ..., \sqrt[3]{n}\}$. Thus $\mathcal{DB}_{000}$ can emulate $\mathcal{DB}_{001}$ by sending the $\sqrt[3]{n}$ bits corresponding to the $\sqrt[3]{n}$ possible queries which could have been sent to $\mathcal{DB}_{001}$. In the same fashion, $\mathcal{DB}_{000}$ can emulate both $\mathcal{DB}_{010}$ and $\mathcal{DB}_{100}$. Thus, by letting $\mathcal{DB}_{000}$ emulate $\mathcal{DB}_{100}$, $\mathcal{DB}_{010}$ and $\mathcal{DB}_{001}$, and letting $\mathcal{DB}_{111}$ emulate $\mathcal{DB}_{011}$, $\mathcal{DB}_{101}$ and $\mathcal{DB}_{110}$, we get a scheme for two databases with total communication complexity $O(\sqrt[3]{n})$. We note that it is too expensive to let $\mathcal{DB}_{000}$ emulate $\mathcal{DB}_{011}$ 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 $i = (i_1, ..., i_d)$, 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 $\mathcal{DB}_c$ ($c \in C_d$) the subcube corresponding to codeword c (i.e., $(S_1^{\sigma_1}, ..., S_d^{\sigma_d})$ where $c = \sigma_1 \cdots \sigma_d$). Each $\mathcal{DB}_c$ replies by emulating itself (i.e., one bit) and the databases corresponding to the words covered by the codeword c (i.e., $n^{1/d}$ bits per each such database). All these answers allow the user to compute the answer it would have received in the protocol for 2^d databases, and consequently retrieve the desired bit. The privacy of the original 2^d-databases scheme is clearly preserved. 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 database and that the total number of bits sent back is $k + (2^d - k) \cdot n^{1/d}$ (note that only the emulation of

dimension (i.e., d)	2^d	# codewords (databases) (i.e., k)	volume (lower) bound	total communication
3	8	2	2	$12n^{1/3}$
4	16	4	4	$28n^{1/4}$
5	32	7	6	$60n^{1/5}$
6	64	12	10	$124n^{1/6}$
7	128	16	16	$224n^{1/7}$
8	256	32	29	$480n^{1/8}$

Figure 1: Covering Codes and Protocols

databases corresponding to non-codewords requires $n^{1/d}$ bits and that it suffices to emulate/cover each such database once[1]). Thus, the total communication equals $(dk + 2^d - k) \cdot n^{1/d} + k$, 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 databases, 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, $k \geq \frac{2^d}{d+1}$ by the *volume bound* (cf., [13]). The 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) [14]. 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 databases) 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 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 [14], 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 databases. 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 databases (i.e., $k = 2$), the communication complexity is $12\sqrt[3]{n} + 2$.
- For four databases (i.e., $k = 4$), the communication complexity is $28\sqrt[4]{n} + 4$.

4 The Polynomial Interpolation Scheme

In this section we describe an information retrieval scheme which requires $O(n^{1/k})$ communication bits for k databases, where $k = O(1)$ is a constant, and $O(\log_2^2 n \log_2 \log_2 n)$ bits where $k = \frac{1}{3} \cdot \log_2 n$. The scheme is based on the method of *low-degree polynomial interpolation*, originating from [6] and extensively used thereafter (see for example [15, 3, 4, 12]). We start by presenting a simple version for $k = \log_2 n + 1$ databases. This version is essentially the one used for $\log_2 n + 1$ oracles in [6]. An improved and more general scheme is developed in Subsection 4.2. This scheme is a variant of the one presented in [7].

4.1 A Simple Scheme For $\log_2 n + 1$ Databases

Supposing that $n = 2^s$, we associate the set $[n] \equiv \{0,1\}^s$ with the set of functions from $[s]$ to $\{0,1\}$. Thus $j \in [n]$ is associated with the function $j : [s] \mapsto \{0,1\}$ so that, for every $\ell = 1, 2, ..., s$, the value $j(\ell)$ is the ℓ-th least significant bit in the binary expansion of j. Let $\delta_{i,j}$ be the Kronecker function:

$$\delta_{i,j} \stackrel{\triangle}{=} \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

We are currently interested in a scheme allowing $\mathcal{U}$ to retrieve the i^{th} bit of $x = x_1 \cdots x_n \in \{0,1\}^n$ using $s+1$ databases. In what follows we shall define a sequence of functions constructed with the value of i in mind.

Let $GF(q)$ be a finite field with at least $s + 2$ elements. Consider a function, defined over $GF(q)$, of the following form:

$$F^{i,x}(z) = \sum_{j \in [n]} f_j^i(z) \cdot x_j$$

where

P1 the f_j^i's are polynomials (in z) of degree at most s; and

P2 $f_j^i(0) = \delta_{j,i}$, for each $j \in [n]$.

By (P2), $F^{i,x}(0) = \sum_{j \in [n]} f_j^i(0) \cdot x_j = \sum_{j \in [n]} \delta_{j,i} \cdot x_j = x_i$. On the other hand, by (P1), $F^{i,x}$ is a polynomial of degree at most s (in z). Thus, if $\mathcal{U}$ is given the value of $F^{i,x}(\cdot)$ at $s+1$ points, it can easily retrieve $F^{i,x}(0)$ by interpolation. So if $\mathcal{U}$ can obtain from each $\mathcal{DB}_p$ ($p = 1, 2, ..., s+1$) the value of F at point $p \neq 0$, without yielding information about i, then we are done. We describe a scheme which achieves this goal at low cost.

The user selects uniformly and independently s elements in the field $GF(q)$, denoted by $r_1, ..., r_s$, and defines s functions

$$g_\ell(z) \stackrel{\triangle}{=} r_\ell \cdot z + i(\ell) \qquad \text{for } \ell = 1, 2, ..., s.$$

[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 $j = 1, ..., k$.

For every $j \in [n]$ and $\ell \in [s]$ we define the degree 1 polynomial

$$f^i_{j,\ell}(z) \triangleq j(\ell) \cdot g_\ell(z) + (1 - j(\ell)) \cdot (1 - g_\ell(z)).$$

The polynomial $f^i_j(z)$ is now defined as the product of the $f^i_{j,\ell}(z)$'s, namely

$$f^i_j(z) = f^i_{j,1}(z) \cdot f^i_{j,2}(z) \cdot \ldots \cdot f^i_{j,s}(z) .$$

The user sends the values $g_1(p), ..., g_s(p)$ to $\mathcal{DB}_p$ (for $p = 1, 2, ..., s+1$). $\mathcal{DB}_p$ uses these s values to compute $F^{i,x}(p)$, even though it does not know i, as follows. First, for every j and ℓ ($1 \leq j \leq n, 1 \leq \ell \leq s$), $\mathcal{DB}_p$ computes the value $f^i_{j,\ell}(p)$ by setting

$$f^i_{j,\ell}(p) = \begin{cases} g_\ell(p) & \text{if } j(\ell) = 1 \\ (1 - g_\ell(p)) & \text{otherwise} \end{cases}$$

Now, for every $j \in [n]$, $\mathcal{DB}_p$ computes

$$f^i_j(p) = f^i_{j,1}(p) \cdot f^i_{j,2}(p) \cdot \ldots \cdot f^i_{j,s}(p)$$

and

$$F^{i,x}(p) = \sum_{j=1}^{n} f^i_j(p) \cdot x_j .$$

This computation takes $O(s \cdot n)$ operations in $GF(q)$. The value $F^{i,x}(p)$ is sent to the user. This way, $\mathcal{U}$ obtains the value of the polynomial $F^{i,x}(z)$ at the $s+1$ points $1, 2, \ldots, \log_2 n + 1$. The user interpolates and obtains $F^{i,x}(0) = x_i$.

We first assert that the scheme provides privacy. This is because the values $g_1(p), ..., g_s(p)$ sent by $\mathcal{U}$ to $\mathcal{DB}_p$ ($p \neq 0$) are uniformly and independently distributed in the field, regardless of i. To see that the scheme yields the correct answer it suffices to verify properties (P1) and (P2) above. By definition, each of the polynomials f^i_j is a product of s linear polynomials, and thus property (P1) follows. Property (P2) holds since $f^i_{j,\ell}(z) = j(\ell) \cdot g_\ell(z) + (1 - j(\ell)) \cdot (1 - g_\ell(z))$ and $g_\ell(0) = i(\ell)$, for each $\ell \in [s]$, and thus

$$\begin{aligned} f^i_j(0) &= \prod_{\ell=1}^{s} f^i_{j,\ell}(0) \\ &= \prod_{\ell=1}^{s} (j(\ell) \cdot i(\ell) + (1 - j(\ell)) \cdot (1 - i(\ell))) \\ &= \delta_{i,j} . \end{aligned}$$

Finally, we consider the communication complexity of the above scheme. The communication between the user and each database consists of s field elements sent from the user to the database and one field element sent in response. Thus, the total communication amounts to $(s+1) \cdot (s+1) \cdot \log_2 q$, where q is the size of the finite field $GF(q)$. This q must be at least $s+2$ (to accommodate $s+1$ non-zero points), and can always be found in the range $[s+2, 2s]$. We have $s = \log_2 n$, so with $s + 1 = \log_2 n + 1$ databases, the communication complexity is $(1 + o(1)) \cdot \log_2^2 n \log_2 \log_2(2n)$.

4.2 The General Case

To handle the general case of k databases, where $k \leq \log_2 n$, we use the same basic idea, but employ a different representation of integers in the range 1 through n. Instead of the "dense" binary representation, we consider s-bit long binary sequences with exactly $k-1$ occurrences of 1 in them[2]. We take the minimum s satisfying $\binom{s}{k-1} \geq n$. Every $1 \leq j \leq n$ is represented by a sequence with $s + 1 - k$ zeroes and $k-1$ ones. We order these sequences in lexicographic order, and represent j by the j-th sequence in this list. We now associate every $j \in [n]$ with a function $j : [s] \mapsto \{0, 1\}$ so that, for every $\ell = 1, 2, ..., s$, the value $j(\ell)$ is the ℓ-th entry in the j-th sequence.

Let $GF(q)$ be a finite field with at least $k+1$ elements. Let i be the bit position which $\mathcal{U}$ wants to retrieve. Again, consider a polynomial $F^{i,x}(z) = \sum_{j \in [n]} f^i_j(z) \cdot x_j$, where

P1 the f^i_j's are polynomials of degree at most $k-1$.

P2 $f^i_j(0) = \delta_{j,i}$, for each $j \in [n]$.

By arguments identical to those used in Subsection 4.1, the value of $F^{i,x}(\cdot)$ at k points enables $\mathcal{U}$ to interpolate and retrieve $x_i = F^{i,x}(0)$. We now describe the protocol.

The user selects uniformly and independently s elements in the field $GF(q)$, denoted by $r_1, ..., r_s$, and defines s functions

$$g_\ell(z) \triangleq r_\ell \cdot z + i(\ell) \qquad \text{for } \ell = 1, 2, ..., s.$$

The user sends the values $g_1(p), ..., g_s(p)$ to $\mathcal{DB}_p$ (for $p = 1, 2, ..., s+1$). For every $j \in [n]$ and $\ell \in [s]$ we define the degree 1 polynomial

$$f^i_{j,\ell}(z) \triangleq j(\ell) \cdot g_\ell(z) + (1 - j(\ell)) \cdot (1 - g_\ell(z)).$$

The next step, however, is different. The analog definition would be to take $f^i_j(z)$ as the product $f^i_{j,1}(z)$ through $f^i_{j,s}(z)$, which is a polynomial of degree s. This would require $s+1$ evaluation points, more than the number of databases we have. We want the polynomial $f^i_j(z)$ to be of degree $k-1$, with $\delta_{j,i}$ as its free term. To achieve this, we define

$$f^i_j(z) = \prod_{\ell : j(\ell)=1} f^i_{j,\ell}(z) .$$

There are exactly $k-1$ indices with $j(\ell) = 1$. Therefore $f^i_j(z)$ is of degree at most $k-1$ (recall that each $f^i_{j,\ell}(z)$ is of degree 1), and so property (P1) holds. Property (P2) holds since

$$\begin{aligned} f^i_j(0) &= \prod_{\ell : j(\ell)=1} f^i_{j,\ell}(0) \\ &= \prod_{\ell : j(\ell)=1} (j(\ell) \cdot i(\ell) + (1 - j(\ell)) \cdot (1 - i(\ell))) \end{aligned}$$

[2] The scheme from [7] is also similar in spirit and can be thought of as using a representation of integers with s-bit long sequences which are divided into $k-1$ blocks of length $s/(k-1)$ and any block having a single 1.

If $j = i$ then the last expression equals

$$\prod_{\ell : j(\ell)=1} \left(i^2(\ell) + (1 - i(\ell))^2 \right) .$$

Each multiplicand equals 1, and therefore the product, $f_i^i(0)$, equals 1. If $j \neq i$, then for at least one ℓ we have $j(\ell) = 1$ and $i(\ell) = 0$ (since both $j(\cdot)$ and $i(\cdot)$ have each exactly $k-1$ entries of value 1). For this ℓ, the multiplicand

$$(j(\ell) \cdot i(\ell) + (1 - j(\ell)) \cdot (1 - i(\ell)))$$

is 0, and therefore the product, $f_j^i(0)$, equals 0.

With this modification, the protocol proceeds similarly to the previous one. The user sends to $\mathcal{DB}_p$ the values $g_1(p), g_2(p), \ldots, g_s(p)$. The arguments for correctness and privacy are the same too. The communication complexity, however, is slightly different. As before, the user sends each database s field elements and receives one field element in response. However, here, the overall communication complexity is $k \cdot (s+1) \cdot \log_2 q \leq k \cdot (s+1) \cdot (1 + \log_2 k)$ (and k is not necessarily equal to $s+1$). Recall that s has to satisfy $\binom{s}{k-1} \geq n$. We get

Theorem 3: Let s, k and n be integers so that $\binom{s}{k-1} \geq n$, and let $q \geq k+1$ be a prime power. Then there exists a private information retrieval schemes for k databases, each holding n bits of data, so that the communication consists of one round in which the user sends $s \cdot \log_2 q$ bits to each database and receives $\log_2 q$ bits in return (from each database).

To exactly analyze the complexity, we separately consider different values of the parameters k and s. One point along this curve is $s = \log_2 n + \log_2 \log_2 n$ and $k = \frac{s}{2} + 1$. Using the approximation $\binom{s}{s/2} \approx \frac{2^s}{\sqrt{s}}$, we get $\binom{s}{k-1} > n$. Thus,

Corollary 4: There are private information retrieval schemes for $\frac{1}{2} \cdot (\log_2 n + \log_2 \log_2 n) + 1$ databases, each holding n bits of data, so that the communication complexity is $\frac{1}{2} \cdot (1 + o(1)) \cdot \log_2^2 n \cdot \log_2 \log_2 (2n)$.

This is less[3] than the communication complexity of the scheme of Subsection 4.1, while the number of databases is slightly over a half of the $\log_2 n + 1$ databases used there. The other extreme on the curve is k constant. Here s is $O(n^{1/(k-1)})$ (actually, $s = (k-1) \cdot \sqrt[k-1]{n+k}$ suffices), and the resulting communication complexity is also $O(n^{1/(k-1)})$ (which is strongly skewed towards the user-to-database direction). This complexity can be brought down to $O(n^{1/k})$, using a generic balancing technique that is presented next.

4.3 A Generic Balancing Technique

Consider an arbitrary scheme for privately retrieving information from several databases in which the communication is carried out in one round (i.e., the user simultaneously queries each database 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). Now, each database views itself as participating in m different executions of the n-bit scheme, each one with a different row (an n-bit string). Namely, in the j-th execution ($j = 1, \ldots, m$), the database 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 database remains unchanged, while the communication in the database-to-user direction increases by a factor of m.

We now apply the balancing technique to the protocol in Theorem 3. To this end, we view the string x as an m-by-(m/n) matrix of bits. Thus, we use the protocol of Theorem 3 for strings of length $\frac{n}{m}$ and so s should now satisfy $\binom{s}{k-1} \geq \frac{n}{m}$.

Theorem 5: Let k, n, m and s be integers so that $\binom{s}{k-1} \geq \frac{n}{m}$ and $q \geq k+1$ be a prime power. Then there exists a private information retrieval schemes for k databases, each holding n bits of data, so that the communication complexity is $k \cdot (m+s) \cdot \log_2 q$.

In particular, setting $s = \sqrt[k-1]{(k-1)! \cdot ((n/m) + k)}$ satisfies the condition $\binom{s}{k-1} \geq \frac{n}{m}$. Setting $m = \sqrt[k]{n}$ (which is not optimal), we get

Corollary 6: Let k and n be integers and $q \geq k+1$ be a prime power. Then there exists a private information retrieval schemes for k databases, each holding n bits of data, so that the communication complexity is

$$k \cdot \left(s + \sqrt[k]{n+k} \right) \cdot \log_2 q$$

where $s \triangleq \sqrt[k-1]{(k-1)!} \cdot \sqrt[k-1]{n^{(k-1)/k} + k}$. We may also use $s = \sqrt[k-1]{(k-1)!} \cdot \sqrt[k]{n} + \sqrt[k-1]{k!}$. Specifically, for $k = 2$ this yields $s = \sqrt{n} + 2$; for $k = 3$, $s = \sqrt{2} \cdot \sqrt[3]{n} + 3$; for $k = 4$, $s = \sqrt[3]{6} \cdot \sqrt[4]{n} + 3$; and for $k \geq 5$ we may use $s = (k-1) \cdot \sqrt[k]{n}$.

This result is asymptotically (for $n \to \infty$) better than the covering codes schemes of Subsection 3.3, except for the cases $k = 2$ and $k = 4$. For $k = 2$, we get here $O(n^{1/2})$ communication, while we had $O(n^{1/3})$ communication there. For $k = 4$, both methods give $O(n^{1/4})$ complexities (and also the constant in the O-notation are comparable[4]).

[3] specifically about one half

[4] Actually, the constant here, $4 \cdot (1 + \sqrt[3]{6}) \cdot \log_2 5 \approx 26.165$, is slightly better than the constant, 28, for the covering codes.

4.4 Further improving the general case

In this subsection we further improve the polynomial interpolation method of subsection 4.2. While this optimization does not improve the asymptotic behavior of the communication for any fixed number of databases, it does achieves significant saving when the number of databases is logarithmic.

As usual, let k denote the number of databases. Again the starting point for the improved scheme is a different representation of integers in the interval from 1 to n. Instead of considering $\{0,1\}$-sequences with $k-1$ occurrences of 1 in them (as in Subsection 4.2), we consider this time sequences of non–negative integers that sum up to exactly $k-1$. We associate with every $j \in [n]$, the function $j : [s] \mapsto \{0,\ldots,k-1\}$, so that for every $\ell = 1,\ldots,s$, the value $j(\ell)$ is the ℓ-th entry in the j-th sequence, and $\sum_{\ell=1}^{s} j(\ell) = k-1$. The number of sequences of length s whose sum equals r is $\binom{s+r-1}{r}$, and hence we will pick the minimum s so that $\binom{s+k-2}{k-1} \geq n$. Let $GF(q)$ be a finite field with at least $k+1$ elements.

Suppose the user $\mathcal{U}$ wishes to retrieve the i-th bit x_i of the database. Let $\hat{\imath}$ be the s-dimensional vector over $GF(q)$ given by $\hat{\imath} = (i(1),\ldots,i(s))$. The user starts by uniformly picking a vector $\hat{w} = (w(1),\ldots,w(s)) \in GF(q)^s$. For $p = 1, ..., k$, the user sends $\hat{\imath} + p\hat{w}$ to $\mathcal{DB}_p$. (Here for two vectors v_1 and v_2, the notation $v_1 + v_2$ is simply the vector sum, and for a scalar a and vector v, the notation av denotes the vector obtained by multiplying each coordinate of v by a.)

Consider a fixed multivariate polynomial $G(\hat{y})$ and a polynomial $F(z) = G(\hat{\imath} + z\hat{w})$ with the following properties:

P1 G is a polynomial on s variables of total degree at most $k-1$ and F is a univariate degree $k-1$ polynomial.

P2 For any $j \in [n]$, $G(\hat{\jmath}) = x_j$, where $\hat{\jmath}$ denotes the vector $(j(1),\ldots,j(s))$. In particular, $F(0) = G(\hat{\imath}) = x_i$.

The database $\mathcal{DB}_p$ responds with the value $F(p) = G(\hat{\imath} + p\hat{w})$. The user views the values $F(1)$, $F(2)$, …, $F(k)$ and interpolates for $F(0)$.

It remains to show that a polynomial G as described above exists. Let $f_{k,j}(\hat{y})$ be the polynomial

$$f_{k,j}(\hat{y}) = \prod_{\ell=1}^{s} \prod_{p=0}^{j(\ell)-1} \frac{y(\ell)-p}{j(\ell)-p}.$$

Then $f_{k,j}(\hat{y})$ has degree $\sum_{\ell=1}^{s} j(\ell) = k-1$ and it satisfies the condition $f_{k,j}(\hat{\jmath}') = \delta_{j,j'}$. (This is obviously true if $j = j'$ as all terms in the above product equal 1; on the other hand, if $j \neq j'$ then since the sum of elements in both vectors $\hat{\jmath}$ and $\hat{\jmath}'$ is the same (i.e., $k-1$) there exists an index ℓ for which $j'(\ell) < j(\ell)$. The term corresponding to this value ℓ and to $p = j'(\ell)$ equals 0 and hence the whole product is 0 as needed.) Now define $G(\hat{y})$ to be $\sum_{j=1}^{n} f_{k,j}(\hat{y})x_j$, where the $f_{k,j}$'s are polynomials of degree at most $k-1$ as above. It is clear that G has degree at most $k-1$. Furthermore, for any $i \in [n]$, $G(\hat{\imath}) = \sum_{j=1}^{n} f_{k,j}(\hat{\imath})x_j = \sum_{j=1}^{n} \delta_{j,i}x_j = x_i$. Thus we obtain the following theorem.

Theorem 7: Let s, k and n be integers so that $\binom{s+k-2}{k-1} \geq n$, and let $q \geq k+1$ be a prime power. Then there exists a private information retrieval schemes for k databases, each holding n bits of data, so that the communication consists of one round in which the user sends $s \cdot \log_2 q$ bits to each database and receives $\log_2 q$ bits in return (from each database).

The improvement provided by the above is quite significant for values of k that are $\Theta(\log_2 n)$. For example, if we take $k = \frac{1}{3} \cdot \log_2 n$ and $s = 2 + \log_2 n$, we have $\binom{s+k-2}{k-1} \approx 2^{H_2(1/4)\cdot\frac{4}{3}\log_2 n} > n^{1.081}$, where $H_2(\cdot)$ is the binary entropy function. This implies,

Corollary 8: There are private information retrieval schemes for $1 + \frac{1}{3}\log_2 n$ databases, each holding n bits of data, so that the communication complexity is $\frac{1}{3}\cdot(1+o(1))\cdot\log_2^2 n \cdot \log_2 \log_2(2n)$.

Employing the balancing technique of Subsection 4.3, we get

Theorem 9: Let k, n, m and s be integers so that $\binom{s+k-2}{k-1} \cdot m \geq n$, and let $q \geq k+1$ be a prime power. Then there exists a private information retrieval schemes for k databases, each holding n bits of data, so that the communication consists of one round in which the user sends $s \cdot \log_2 q$ bits to each database and receives $m \log_2 q$ bits in return (from each database).

For fixed k and growing $n \to \infty$, this result leaves the asymptotic communication complexity as it was, $O(n^{1/k})$. Similarly, for relatively small k (w.r.t. n) this result has little effect on the actual numbers. However, for relatively larger k (for example $k = 16$ and $n = 2^{40}$) the saving is meaningful. The next section provides a sample of numeric results.

5 Numeric Results

Figure 2 summarizes the communication costs required for private retrieval of a single data bit. We include a sample of databases numbers ($k = 2, 4, 7, 16$)[5] and sizes ($n = 2^{20}, 2^{30}, 2^{40}$). For the polynomial interpolation method, we included both the "basic" results and the improved ones. (Except the case $k = 2$, where s and m satisfy $(s+1)m \geq n$ in the improved method, instead of $sm \geq n$, which results in a meaningless improvement). It is readily seen from this table that for $k = 2$ the covering codes method is superior to the polynomial interpolation method. For $k = 4$, $k = 7$ and $k = 16$, the polynomial interpolation method is superior, especially for large values of n. For the interpolations schemes the asymptotic expression is obtained by setting $m = \sqrt[k]{n}$ (as in Corollary 6) whereas the actual figures in the last three columns are obtained

[5] Except for $k = 12$, these are the only values for which covering-code schemes exist. For other values of k one can only utilize k' databases, where $k' < k$ is the largest integer for which a covering code exists. For example, to get a scheme for $k = 6$ databases, we use $k' = 4$.

number of databases	method	communication complexity	total communication bits		
			$n = 2^{20}$	$n = 2^{30}$	$n = 2^{40}$
$k = 2$	covering codes	$12 \cdot \sqrt[3]{n}$	1,224	12,300	123,864
$k = 2$	polynomial interpolation	$6.34 \cdot \sqrt[2]{n}$	6,493	207,745	6,647,815
$k = 4$	covering codes	$28 \cdot \sqrt[4]{n}$	924	5,096	28,700
$k = 4$	polynomial interpolation	$26.17 \cdot \sqrt[4]{n}$	827	4,635	26,136
$k = 4$	*improved* interpolation	as above	809	4,616	26,118
$k = 7$	covering codes	$60 \cdot \sqrt[5]{n}$	1,020	3,900	15,420
$k = 7$	polynomial interpolation	$83.87 \cdot \sqrt[7]{n}$	651	1,638	4,326
$k = 7$	*improved* interpolation	as above	546	1,533	4,221
$k = 16$	covering codes	$224 \cdot \sqrt[7]{n}$	1,792	4,480	11,872
$k = 16$	polynomial interpolation	$485.5 \cdot \sqrt[16]{n}$	1,635	2,224	3,205
$k = 16$	*improved* interpolation	as above	720	1,308	2,289

Figure 2: Comparison of some concrete schemes

via a better (optimized) choice of m (which is slightly larger than $\sqrt[k]{n}$). For both versions of the polynomial interpolation scheme, for $k = 2, 4, 7, 16$ we use finite fields with $q = 3, 5, 8, 17$ elements, respectively. Except $q = 8$, these are not powers of 2. To encode them efficiently by binary strings we pack multiple letters together. Thus we can represent $GF(q)$ elements by using $\log_2 q$ bits, which was used in the table (representation by $\lceil \log_2 q \rceil$ bits cause a substantial degradation).

Finally, we remark that when privately retrieving a larger block of data (for example, 2^{10} consecutive bits), the resulting communication is smaller than simply the communication for single bit multiplied by the block size. This is achieved by using the balancing technique, tuned to the block size. The improvement is applicable for both the covering codes and the polynomial interpolation method. Thus, the incurred overhead, compared to non-private retrieval, is substantially smaller for this more realistic case than for the single bit case. For example, the improved interpolation method for $k = 4$ databases communicates $11,238$ bits for retrieving a block of 2^{10} bits from a database of size $n = 2^{30}$ (bits), and $26,768$ bits for retrieving the same block from a database with $n = 2^{40}$. So the communication overhead is about 11 and 26 bits per one information bit, respectively. For further discussions see Section 6.

6 Private Information Retrieval of Blocks

In this section we consider a more realistic model of private information retrieving in which the information is partitioned into blocks (or records) rather than single bits. For simplicity, we assume that each block/record contains ℓ bits. We denote by $\mathcal{PIR}(\ell, n, k)$ the problem of retrieving privately an (ℓ-bit long) information block from k databases, each holding the same n blocks. Previous sections have dealt with $\mathcal{PIR}(1, n, k)$. Clearly $\mathcal{PIR}(\ell, n, k)$ can be solved by ℓ invokations of $\mathcal{PIR}(1, n \cdot \ell, k)$,[6] but there are much more efficient reductions of $\mathcal{PIR}(\ell, \cdot, k)$ to $\mathcal{PIR}(1, \cdot, k)$.

We start by noting that the Generic Balancing Technique of Section 4.3 actually provides such a reduction. Specifically,

Proposition 10: Suppose that $\mathcal{PIR}(1, n, k)$ can be solved by a one-round protocol in which the user sends $\alpha_k(n)$ bits to each database and receives $\beta_k(n)$ bits in return (from each database). Then, for every $\ell > 1$, $\mathcal{PIR}(\ell, n, k)$ can be solved by a one-round protocol in which the user still sends $\alpha_k(n)$ bits to each database and receives $\ell \cdot \beta_k(n)$ bits in return (from each database).

In Section 4.3 we emphasized the asymmetric effect that the above transformation has on the communication complexity (i.e., increasing the communication from the databases to the user while maintaining the communication complexity in the other direction). An "asymmetric" transformation of the opposite flavour follows.

Proposition 11: Suppose that $\mathcal{PIR}(1, n, k)$ can be solved by a one-round protocol in which the user sends $\alpha_k(n)$ bits to each database and receives $\beta_k(n)$ bits in return (from each database). Furthermore, suppose that the user retreives the desired information bit by computing $g(\sum_{j=1}^{k} f_j(\gamma_j))$, where γ_j is the message obtained from $\mathcal{DB}_j$, the f_j's are arbitrary fixed functions mapping binary strings into elements of some finite field (of cardinality at most $2^{\beta_k(n)}$), summation is done over this field and g is an homomorphism of the field onto $GF(2)$. (We stress that the f_j's may not depend on the desired bit nor on the randomness used by $\mathcal{U}$.) Then, for every $m > 1$, $\mathcal{PIR}(1, m \cdot (n-1), k)$ can be solved by a one-round protocol in which the user sends $m \cdot \alpha_k(n)$ bits to each database and receives $\beta_k(n)$ bits in return (from each database).

We note that all "pure" schemes (i.e., before "optimizing via balancing") presented in previous sections meet the hypothesis of the proposition. Furthermore, the proposition can be generalized to $\mathcal{PIR}(\ell, \cdot, k)$ schemes (in which each bit in the block is computed as conditioned above).

[6] In fact, $\mathcal{PIR}(\ell, n, k)$ can be easily solved by ℓ invokations of $\mathcal{PIR}(1, n, k)$ just by considering in the j-th invocation only the j-th bit of each of the n blocks.

Proof: For simplicity, we first assume that the f_j's mentioned in the hypothesis are identity transformations. Our solution to $\mathcal{PIR}(1, m \cdot (n-1), k)$ follows. We partition the $N \stackrel{\triangle}{=} m \cdot (n-1)$ bits, in each database, into m strings each holding $n-1$ bits and augment each of these strings by a dummy position set to zero. Bit positions in $[N]$ are represented as pairs in $[m] \times [n-1]$ in the natural manner. The user, wishing to retreive $i = [i', i''] \in [m] \times [n-1]$, employs $\mathcal{PIR}(1, n, k)$ in parallel m times. In the j^{th} instance $\mathcal{U}$ behaves as when asking for position i'' if $j = i'$ and as asking for position n otherwise. Each database adds together the answers it would have sent in each of the m invokations of $\mathcal{PIR}(1, n, k)$ and sends this sum as its only message. The user just adds all answers it has obtained and applies g as it would have done in a single invokation of $\mathcal{PIR}(1, n, k)$. We stress that each database sends only one $\beta_k(n)$-bit long string rather than m such strings. Evidently, the new scheme satisfies the privacy requirement. Correctness follows from associativity of addition, the hypothesis that g is a homomorphism, and the fact that the dummy position (i.e., position n) is set to 0. Specifically, let γ_p^j be the designated answer of $\mathcal{DB}_p$ in the j^{th} invokation. We know that $g(\sum_p \gamma_p^j)$ equals $x_i \in \{0, 1\}$ if $j = i'$ and 0 otherwise. Thus,

$$g\left(\sum_p \sum_j \gamma_p^j\right) = \sum_j g\left(\sum_p \gamma_p^j\right) = x_i$$

It is left to extend the protocol to the general case. We first observe that the hypothesis regrading the field size allows us to modify the original $\mathcal{PIR}(1, n, k)$ protocol so that all the f_i's are identity transformation. □

Combining the above two propositions, we obtain.

Corollary 12: Let $\mathcal{PIR}(1, n, k)$ be as in Proposition 11 and $\ell, m > 1$. Then, $\mathcal{PIR}(\ell, m \cdot (n-1), k)$ can be solved by a one-round protocol in which the user sends $m \cdot \alpha_k(n)$ bits to each database and receives $\ell \cdot \beta_k(n)$ bits in return (from each database). In particular, $\mathcal{PIR}(\ell, n, k)$ can be solved within ℓ times the complexity of $\mathcal{PIR}(1, \frac{n}{\ell} + 1, k)$.

In some settings the number of records is not substaintially 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 retreival). Furthermore, for $\ell \geq n$, the basic two-databases scheme (of Section 3.1) becomes of interest. Specifically, we get an overhead factor of four:

Corollary 13: Let $\ell \geq n$. Then, $\mathcal{PIR}(\ell, n, 2)$ can be solved by a one-round protocol of total communication complexity $4 \cdot \ell$.

Proof: We use the $\mathcal{PIR}(1, n, 2)$ scheme (of Section 3.1) in which $\mathcal{U}$ sends n bits to each database (indicating a linear combination of the bits in the database) and xor-s (i.e., adds over $GF(2)$) the answers it obtains. Using Proposition 10, we are done. □

Note that unlike the $\mathcal{PIR}(1, n, 2)$ scheme (of Section 3.1), the obvious $\mathcal{PIR}(1, n, 2)$ (or actually $\mathcal{PIR}(1, n, 1)$), in which each database sends its contents to the user who then retreives the desired bit, is not an adequate starting point for applying Proposition 10.

7 Privacy With Respect to Coalitions

Our results so far concerned the privacy of the user with respect to any *single* database. It is not hard to verify that in all the schemes described so far, any two databases get some information about the desired index i from their joint queries (and in some of the schemes can even recover it). In this section we consider the scenario where the goal is to guarantee the privacy of the user with respect to any *coalition* of no more than t databases. The definition of privacy follows the one given in Section 2 but considers the joint probability distribution of communication seen by any $t' \leq t$ databases. We present a modification of the protocol in Subsection 4.2 to this scenario.

Given the parameter t (maximum number of databases in a coalition) and n (input length), let k and s satisfy $\binom{s}{k-1} \geq n$. The number of databases we use here is $t(k-1)+1$ (for $t=1$ this gives k as in 4.2). We consider the same representation of numbers $j \in [n]$ as sequences of length s with $k-1$ ones and $s-k+1$ zeroes. Let $GF(q)$ be a finite field with at least $t(k-1)+1$ elements. Let x_i be the bit $\mathcal{U}$ wants to retrieve. Again, consider a polynomial

$$F^{i,x}(z) = \sum_{j \in [n]} f_j^i(z) \cdot x_j$$

where now

P1 the f_j^i's are polynomials of degree at most $t(k-1)$.

P2 $f_j^i(0) = \delta_{j,i}$, for each $j \in [n]$.

By arguments which are familiar by now, the value of $F^{i,x}(\cdot)$ at $t(k-1)+1$ points enables $\mathcal{U}$ to interpolate and retrieve $x_i = F^{i,x}(0)$.

The first part of the protocol is different. The user selects s random independent polynomials of degree t in $GF(q)$, where the free term of the ℓ-th polynomial, $g_\ell(z)$, is $i(\ell)$ ($\ell = 1, 2, \ldots, s$). The user sends the values $g_1(p), \ldots, g_s(p)$ to $\mathcal{DB}_p$ (for $p = 1, 2, \ldots, s+1$). For every $j \in [n]$ and $\ell \in [s]$ we define the degree-t polynomial

$$f_{j,\ell}(z) \stackrel{\triangle}{=} j(\ell) \cdot g_\ell(z) + (1 - j(\ell)) \cdot (1 - g_\ell(z)).$$

The definition of the polynomials $f_j(z)$, as well as rest of the protocol, is identical to 4.2, and we will not repeat it.

It is clear that the new property (P1) holds, and (P2) is unchanged. This proves the correctness of the protocol. To show that it is t-private, we observe that, for any ℓ and any t non-zero points in $GF(q)$, the values of the polynomial $g_\ell(\cdot)$ at these t points are uniformly distributed in $GF(q)$. Furthermore, the values of the s polynomials $g_\ell(z)$ ($\ell = 1, 2, \ldots, s$) at these t points are independent, and so any t databases receive $t \cdot s$ values that are uniformly distributed in $GF(q)$.

The communication complexity is as in Subsection 4.2: the user sends each database s field elements and receives one field element in response. The number of databases is $t(k-1)+1$, so the overall communication complexity is $(t(k-1)+1) \cdot (s+1) \cdot \log_2 q$. The balancing techniques of Subsection 4.3 can be applied here as well. The straightforward details are omitted. To summarize,

Theorem 14:

- Let t and d be be integer functions and $c > 1$ be a constant so that $d(n) = c \cdot t(n)$. Then there are $t(\cdot)$-private information retrieval schemes for $d(\cdot)$ databases, in which the communication complexity is $O(t(n) \cdot \sqrt[c]{n})$.
- Let t be an integer function of n and $d(n) = t(n) \cdot \log_2 n$. Then there are $t(\cdot)$-private information retrieval schemes for $d(\cdot)$ databases with communication complexity $\mathrm{polylog}(n) \cdot t(n)$.

We comment that the latter result is about the best we can hope given our state of knowledge with respect to 1-private schemes (e.g., Corollary 6). This is because of

Proposition 15: Let $1 < t < d$. The communication complexity of a t-private information retrieval scheme for d databases is at least as the communication complexity of a 1-private information retrieval scheme for $\lceil d/t \rceil$ databases.

Proof sketch Given a t-private scheme for d databases we construct a 1-private scheme for $\lceil d/t \rceil$ databases by letting each database in the new scheme emulate t databases in the original scheme.

8 Conclusions and Open Problems

We have presented several techniques for constructing private (i.e., 1-private) information retrieval schemes. Our feeling is that the scheme for 2 databases is essentially the best one can hope for (with respect to communication complexity). More generally, we conjecture that private information retrieval schemes for a constant number, k, of databases, each holding n bits of data, require $\Omega(\sqrt[k+1]{n})$ communication. Even if this conjecture is true, it leaves a gap towards our k database schemes which, for $k > 2$, use $O(\sqrt[k]{n})$ communication.

In an attempt to develop lower bound for the problem, we considered the very simple case in which there are two databases and the user makes a single *binary* 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.

Acknowledgment

We wish to thank Muli Safra, Shafi Goldwasser, Don Coppersmith and Joe Kilian for helpful discussions regarding related issues. We are grateful to Tuvi Etzion for providing us some pointers to the known results on covering codes, 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] L. BABAI, L. FORTNOW, AND C. LUND. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, v. 1, pp. 3-40, 1991.

[4] L. BABAI, L. FORTNOW, L. LEVIN, AND M. SZEGEDY. Checking computations in polylogarithmic time. STOC, 1991.

[5] L. BABAI, P. KIMMEL, AND S. V. LOKAM. Simultaneous Messages vs. Communication. To appear *STACS*, 1995.

[6] D. BEAVER AND J. FEIGENBAUM. Hiding Instances in Multioracle Queries. *STACS*, 1990.

[7] D. BEAVER, J. FEIGENBAUM, J. KILIAN AND P. ROGAWAY. Security with Low Communication Overhead. *CRYPTO*, 1990.

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

[11] 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.

[12] U. FEIGE, S. GOLDWASSER, L. LOVASZ, S. SAFRA, AND M. SZEGEDY. Approximating clique is almost NP-Complete. *FOCS*, 1991.

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

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

[15] C. LUND, L. FORTNOW, H. KARLOFF, AND N. NISAN. Algebraic Methods for Interactive Proof Systems. *FOCS*, 1990.

[16] P. PUDLÁK, AND V. RÖDL. Modified Ranks of Tensors and the Size of Circuits. *STOC*, 1993.

[17] 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.

[18] P. TENDICK, AND N. MATLOFF. A Modified Random Perturbation Method for Database Security. *ACM Transactions on Database Systems*, 19:1, pp. 47—63, 1994.

[19] J. D. ULLMAN. Principles of Database Systems. Second edition, 1982.