Robust Information-Theoretic Private Information Retrieval*

Amos Beimel[†] Yoav Stahl[‡]

July 11, 2006

Abstract

An information-theoretic private information retrieval protocol allows a user to retrieve a data item of its choice from a database replicated amongst several servers, such that each server gains absolutely no information on the identity of the item being retrieved. One problem with this approach is that current systems do not guarantee availability of servers at all times for many reasons, e.g., crash of server or communication problems. In this work, we design robust PIR protocols, i.e., protocols which still work correctly even if only some servers are available during the protocols' operation. We present various robust PIR protocols giving different tradeoffs between the different parameters. We first present a generic transformation from regular PIR protocols to robust PIR protocols. We then present two constructions of specific robust PIR protocols. Finally, we construct robust PIR protocols which can tolerate Byzantine servers, i.e., robust PIR protocols which still work in the presence of malicious servers or servers with corrupted or obsolete database.

*A preliminary version of this paper appeared in [13].
[†]Computer Science Dept., Ben-Gurion University, Beer-Sheva 84105, Israel.
[‡]Computer Science Dept., Ben-Gurion University, Beer-Sheva 84105, Israel.

1 Introduction

A Private Information Retrieval (PIR) protocol allows a user to retrieve a data item of its choice from a database, such that the server storing the database does not gain information on the identity of the item being retrieved. For example, an investor might want to know the price of a certain stock in the stock-market without revealing which stock she is interested in. The problem was introduced by Chor, Goldreich, Kushilevitz, and Sudan [23], and has attracted a considerable amount of attention. It is convenient to model the database by an n-bit string x, where the user, holding some *retrieval index* i, wishes to learn the i-th data bit x_i. In the information-theoretic setting, the user accesses replicated copies of the database kept on different servers, requiring that each server gains absolutely no information on the bit the user reads.

The definition of PIR protocols raises a simple question – what happens if one of the servers crashes during the operation? How can we devise a protocol which still works in the presence of crashing servers? Current systems do not guarantee availability of servers at all times for many reasons, e.g., crash of server or communication problems. Our purpose is to design robust PIR protocols. Given a database x which is replicated amongst ℓ servers, and a parameter $k \leq \ell$ which specifies the minimal number of servers that are available at any moment, the user in our protocol can retrieve x_i by using the answers of any k servers. I.e., even if ℓ-k severs are unreachable while the protocol is being performed, the user can still reconstruct x_i. The user does not need to know in advance which servers are online and which servers will be online during the process.

A trivial solution to this problem is to execute an independent PIR protocol for each group of k servers. This yields a solution whose complexity is $\binom{\ell}{k}$ times the complexity of the best known PIR protocol. Even for fairly small ℓ and k, the factor $\binom{\ell}{k}$ may be too expensive. Another trivial solution is that the user first checks which servers are available and then executes a regular PIR protocol with these servers. The problem with this solution is that it necessitates two rounds of communication. Another problem is that servers can crash between the first round and the second round. Our goal is to design robust protocols in which the dependency of the communication complexity on ℓ and k is polynomial.

We next present two additional motivating examples. First, consider a database which is updated frequently. In this case, the servers might hold different versions of the database. If the user and servers execute a robust PIR protocol, and each server sends the version number of the database, then as long as a big enough subset of the servers holds the latest version of the database, the user can recover the desired bit. Second, consider a system in which the servers do not have the same response time. Furthermore, the response time may vary according to the server's load at a specific moment. In this case, using a robust protocol, the user needs only the first k answers it receives, i.e., it need not wait for slow servers.

1.1 Related Work

Before proceeding, we give an overview of some known results on PIR. The simplest solution to the PIR problem is sending the entire database to the user. This solution is impractical for large databases. However, if a single server is not allowed to gain *any* information about the retrieved bit, then the linear communication complexity of this solution is optimal [23]. To overcome this problem, Chor et al. [23] suggested that the user accesses replicated copies of the database kept on different servers, requiring that each server gains absolutely no information on the bit the user reads (thus, these protocols are called *information-theoretic* PIR protocols). The best information-theoretic PIR protocols known to date are summarized below:

1. A 2-server protocol with communication complexity of $O(n^{1/3})$ [23],

2. A k-server protocol, for any constant $k > 1$, with communication complexity of $O(k^2 \log k n^{1/(2k-1)})$ bits [65] (improving on [23, 3, 38, 37, 8]),

3. A k-server protocol, for any constant $k > 1$, with communication complexity of $O(2^{\tilde{O}(k)} \cdot n^{\frac{2 \log \log k}{k \log k}})$ bits [11], and

4. A protocol with $O(\log n)$ servers and communication complexity of $O(\log^2 n \log \log n)$ bits [5, 6, 23] (the protocol of [5, 6] was done in the context of the instance hiding problem).

In all these protocols it is assumed that the servers do not communicate with each other. Protocols in which the user is protected against collisions of up to t servers, called t-private protocols, have been considered in [23, 37, 8, 65]. Specifically, the best communication complexity of such a protocol is $O(k^2/t \log k n^{1/\lfloor(2k-1)/t\rfloor})$ bits [65]. No strong general lower bounds on PIR are known. A constant-factor improvement over the trivial $\log n$ bound, for any constant k, was obtained in [47]. The constant in the lower bound for 2-server PIR was improved to $5 \log n$ in [64]. Stronger lower bounds were given for restricted protocols [39, 36, 7, 41, 64, 54]. Several extensions of the basic model of PIR have been considered in [12, 26, 34, 35, 50]. A survey on private information retrieval can be found in [32].

One particularly interesting application of PIR is for the construction of so-called *locally decodable codes*. A locally decodable code allows encoding a database x into a (longer) string y, such that even if a large fraction of y is adversarially corrupted, each bit of x can still be decoded *with high probability* by probing few (randomly selected) locations in y. In [40], tight connections between such codes and information-theoretic PIR has been shown. In particular, information-theoretic PIR protocols can be converted into locally decodable codes of related efficiency, and the best known upper bounds on the length of locally decodable codes were obtained from PIR protocols. Lower bounds for locally decodable codes were introduced in [40, 36, 52, 25, 41, 64, 57].

A different approach for reducing the communication is to limit the power of the servers; i.e., to relax the perfect privacy requirement into *computational indistinguishability* against computationally bounded servers (thus, these protocols are called *computational* PIR protocols). In [43], following a 2-server protocol of [22], it is proved that in this setting one server suffices; under a standard number-theoretic intractability assumption they construct, for any constant $\varepsilon > 0$, a *single* server protocol with communication complexity of $O(n^{\varepsilon})$ bits. Essentially the same construction can be based on any homomorphic encryption scheme [47, 60, 66]. A single server protocol with polylogarithmic communication complexity, based on a new number-theoretic intractability assumption called *the Φ-hiding assumption*, is presented in [19]. Other single server protocols with polylogarithmic communication complexity, based on different hardness assumptions, were presented in [21, 45, 33]. In [44], a construction of a protocol based on a very general assumption, the existence of trapdoor permutation, with communication complexity $n\left(1 - \frac{1}{\mathrm{polylog}(n)}\right)$ is presented. Necessary conditions for the existence of computational PIR protocols with sub-linear communication were presented in [10, 27]. Other works which use PIR protocols are [53, 20, 28].

One of the main tools we use in this paper is *perfect hash families* which were introduced by Sprugnoli [59]. These families are used to construct a data structure enabling a retrieval of an item from a static table with a single probe. Several constructions of perfect hash families were given, e.g., [59, 63, 31, 58]. In the last few years, perfect hash families have been applied to circuit complexity problems [51], derandomization of probabilistic algorithms [2], threshold cryptography [14, 16], and other tasks in cryptography [29, 61]. Perfect hash families are also considered from a combinatorial point of view [1, 4, 15, 17, 30, 42, 62]. A comprehensive overview on perfect hashing can be found in [24].

1.2 Our Results

We present several protocols with various features which address the robust PIR problem. These protocols are incomparable, i.e., for different values of n and k we will get better results using different protocols.

Our first result is a generic transformation from k-out-of-k PIR protocols to robust k-out-of-ℓ PIR protocols: we show that if there exists an (ℓ, k) minimal perfect hash family of size $w_{\ell,k}$ (for the definition of minimal perfect hash families, see Definition 3.3) and if there exists a k-out-of-k PIR protocol with communication complexity $\mathrm{PIR}_k(n)$ per server, then there exists a k-out-of-ℓ PIR protocol with communication complexity $w_{k,\ell} \cdot \mathrm{PIR}_k(n)$ per server. Since this transformation is generic, any improvement in the communication complexity of k-out-of-k PIR protocols (e.g., the result of [11]) directly translates to improved robust PIR protocols. The best known explicit constructions of hash families [48, 58] have size $\log \ell \cdot 2^{O(k)}$ (this is basically optimal [48]). That is, this transformation is logarithmic in ℓ, however it is exponential in k. We also present a generic transformation from t-private k-out-of-k PIR protocols to robust t-private k-out-of-ℓ PIR protocols.

Our second result is a robust PIR protocol using the polynomial interpolation based PIR protocol of [5, 6, 23]. This protocol is a k-out-of-ℓ PIR protocol with communication complexity of $O(kn^{\frac{1}{k}} \ell \log \ell)$. That is, the communication in this protocol is polynomial in ℓ and k. However its dependency on n is worse than the protocols obtained via the generic transformation.

Our third protocol combines Shamir's secret sharing scheme with the 2-server protocol of [23]. This results in a 2-out-of-ℓ protocol with communication complexity of $O(n^{1/3} \ell \log \ell)$, that is, the same communication complexity that can be achieved using the generic protocol. We present this protocol as it is a more direct approach; we hope that this approach will be used in the future to construct more efficient protocols for larger values of k.

Finally, we extend our discussion to robust PIR protocols which can tolerate Byzantine servers. That is, we require that the user can reconstruct the correct value of x_i even if the answers of some servers are maliciously altered. We first show a generic transformation from robust PIR protocols to robust PIR protocols that tolerate Byzantine servers. In particular, we obtain two robust k-out-of-ℓ PIR protocols where the user can reconstruct the correct value of x_i as long as it receives at least k answers of which at most $k/3$ are corrupted with communication complexity $2^{O(k)} n^{1/(2\lfloor k/3 \rfloor - 1)} \ell \log \ell$ and $2^{\tilde{O}(k)} n^{O(\log \log k / k \log k)} \ell \log \ell$. More generally, our protocols exhibit a tradeoff between the number of Byzantine servers and the communication complexity. We next present an explicit construction of a robust k-out-of-ℓ PIR protocol where the user can reconstruct the correct value of x_i as long as it receives at least k answers of which at most $k/3$ are corrupted with communication complexity $O(kn^{1/\lfloor k/3 \rfloor} \ell \log \ell)$ (that is, better dependency on k but worse dependency on n).

We summarize the complexity of the various protocols we obtain in Table 1.

1.3 Subsequent Work

In parallel to our work [13], the question of private information retrieval in the presence of Byzantine failures has been addressed also in [67]. They construct a b-private b Byzantine-robust k-out-of-ℓ PIR protocol (where $b < k/2$) with communication $O(n\ell \log \ell)$ (for the definition of b-private b Byzantine-robust PIR see Section 6). Using a b-private k-out-of-k protocol of [23] combined with the ideas of [67], this can be improved to $O(\frac{k}{b} n^{1/(\lfloor k/2b \rfloor + 1)} \ell \log \ell$. In comparison, our protocol, presented in Theorem 6.6, is slightly more efficient and has communication complexity $O(\frac{k}{b} n^{\frac{1}{\lfloor (k-2)/b \rfloor - 1}} \ell \log \ell)$, however it requires that $b < k/3$.

PIR Type	Complexity	Method	Where
k-out-of-ℓ	$2^{O(k)}n^{\frac{1}{2k-1}}\ell\log\ell$	Generic + [58, 48, 3]	Cor. 3.6
k-out-of-$-\ell$	$2^{\tilde{O}(k)}n^{\frac{2\log\log k}{k\log k}}\ell\log\ell$	Generic + [58, 48, 11]	Cor. 3.7
k-out-of-ℓ	$O\left(kn^{\frac{1}{k}}\ell\log\ell\right)$	Polynomial Interpolation	Thm. 4.3
2-out-of-ℓ	$O\left(n^{1/3}\ell\log\ell\right)$	Shamir's Secret Sharing	Thm. 5.2
t-private k-out-of-ℓ	$O\left(2^{O(k)}n^{1/\lfloor(2k-1)/t\rfloor}\ell\log\ell\right)$	Generic + [58, 48, 8]	Cor. 3.9
t-private k-out-of-ℓ	$O\left(\frac{k}{t}n^{\frac{1}{\lfloor(k-1)/t\rfloor+1}}\ell\log\ell\right)$	Polynomial Interpolation	Thm. 4.5
$\left\lfloor\frac{k-a}{2}\right\rfloor$ Byz. k-out-of-ℓ	$2^{O(a)}n^{1/(2a-1)}\ell\log\ell$	Generic + Cor. 3.6	Cor. 6.3
$\left\lfloor\frac{k-a}{2}\right\rfloor$ Byz. k-out-of-ℓ	$2^{\tilde{O}(a)}n^{\frac{2\log\log a}{a\log a}}\ell\log\ell$	Generic + Cor. 3.7	Cor. 6.3
$\left\lfloor\frac{k}{3}\right\rfloor$ Byz. k-out-of-ℓ	$O\left(kn^{1/\lfloor k/3\rfloor}\ell\log\ell\right)$	Polynomial Interpolation	Thm. 6.4
b-private b Byz. k-out-of-ℓ ($b < k/3$)	$O\left(\frac{k}{b}n^{1/(\lfloor(k-2)/b\rfloor-1)}\ell\log\ell\right)$	Generic + Thm. 4.5	Thm. 6.6

Table 1: Summary of the complexity of our various protocols.

In a work done after our original work, Woodruff and Yekhanin [65] suggested an elegant *geometric* approach to information-theoretic PIR. Using their approach, they obtain several improvements over previous protocols in the dependency in k. Their results are already surveyed in Section 1.1; our results do not rely on the results of [65]. In addition, Woodruff and Yekhanin [65] improve our results in Corollary 3.6 and Theorem 4.3, presenting a k-out-of-ℓ PIR protocol with complexity $O(kn^{1/(2k-1)}\ell\log\ell)$. The communication complexity of this protocol is incomparable to the communication complexity of our protocol presented in Corollary 3.7. Plugging the result of [65] in Theorem 6.2, we obtain a $\lfloor(k-a)/2\rfloor$ Byzantine-robust k-out-of-ℓ PIR protocols with total communication $O(an^{1/(2a-1)}\ell\log\ell)$.

Organization. In Section 2 we provide the necessary definitions. In Section 3 we describe our generic transformations from PIR protocols to robust PIR protocols. In Section 4 and in Section 5 we describe specific constructions of robust PIR protocols. In Section 6 we present a robust PIR protocol tolerating Byzantine servers. In Section 7 we discuss some open problems. Finally, in Appendix A we describe a construction of the perfect hash family we use in our paper.

2 Preliminaries

2.1 Notation

We start with some notation used throughout the papers. The set $\{1,\ldots,k\}$ is denoted by $[k]$. The finite field with q elements, where q is a prime-power, is denoted by GF(q). Vectors are denoted by bold letters, e.g., $\mathbf{V}$. The j-th coordinate of a vector $\mathbf{V}$ is denoted by $V[j]$.

2.2 PIR Protocols

We define 1-round information-theoretic PIR protocols.[1] A k-out-of-ℓ PIR protocol involves ℓ servers $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$, each holding the same n-bit string x (the database), and a user who wants to retrieve a bit x_i of the database.

Definition 2.1 (Robust PIR) *A* robust t-private k-out-of-ℓ PIR protocol $\mathcal{P} = (\mathcal{R}, \mathcal{Q}, \mathcal{A}, \mathcal{C})$ *consists of a probability distribution $\mathcal{R}$ over a given set R (the set R is part of the specification of the protocol) and three algorithms: query algorithm $\mathcal{Q}(\cdot,\cdot,\cdot)$, answering algorithm $\mathcal{A}(\cdot,\cdot,\cdot)$, and a reconstruction algorithm $\mathcal{C}(\cdot,\cdot,\ldots,\cdot)$ ($\mathcal{C}$ has $k+3$ arguments). At the beginning of the protocol, the user picks a random string r according to the distribution $\mathcal{R}$. For $j = 1, \ldots, \ell$, it computes a query $q_j = \mathcal{Q}(j, i, r)$ and sends it to server $\mathcal{S}_j$. Each server responds with an answer $a_j = \mathcal{A}(j, q_j, x)$. (The answer is a function of the query and the database; without loss of generality, the servers are deterministic.) Finally, the user, upon receiving (at least) k answers $a_{j_1}, \ldots, a_{j_k}$, computes the bit x_i by applying the reconstruction algorithm $\mathcal{C}(i, r, K, a_{j_1}, \ldots, a_{j_k})$, where $K = \{j_1, \ldots, j_k\}$. The protocol must satisfy the following requirements:*

Correctness. *The user always computes the correct value of x_i from any k answers. Formally, for every $i \in \{1, \ldots, n\}$, every string $r \in R$, every set $K = \{j_1, \ldots, j_k\} \subseteq \{1, \ldots, \ell\}$, and every database $x \in \{0,1\}^n$,*

$$\mathcal{C}(i, r, K, \mathcal{A}(j_1, \mathcal{Q}(j_1, i, r), x), \ldots, \mathcal{A}(j_k, \mathcal{Q}(j_k, i, r), x)) = x_i.$$

t-Privacy. *Each collusion of (at most) t servers has no information about the bit that the user tries to retrieve: For every two indices $i_1, i_2 \in \{1, \ldots, n\}$, for every $\{j_1, \ldots, j_t\} \subseteq \{1, \ldots, \ell\}$, and for every t possible queries $\{\varrho_1, \ldots, \varrho_t\}$*

$$\Pr[\forall_{b \in [t]} \mathcal{Q}(j_b, i_1, r) = \varrho_b] = \Pr[\forall_{b \in [t]} \mathcal{Q}(j_b, i_2, r) = \varrho_b],$$

where the probability is taken over the choice of $r \in R$ according to the distribution $\mathcal{R}$.[2]

We refer to a robust t-private k-out-of-ℓ PIR protocol as a t-private k-out-of-ℓ PIR protocol and to a robust 1-private k-out-of-ℓ PIR protocol as a k-out-of-ℓ PIR protocol. The main difference between a PIR protocol and a robust PIR protocol is in the correctness requirements. That is, the regular k-server PIR protocols are robust k-out-of-k PIR protocols.

Definition 2.2 (Communication Complexity) *Given a k-out-of-ℓ PIR protocol, the* communication per server *is the number of bits communicated between the user and any single server on a database of size n, maximized over all choices of $x \in \{0,1\}^n, i \in \{1, \ldots, n\}$, and random inputs. The* total communication *in the protocol is the number of bits communicated between the user and the ℓ servers. The* query complexity *(per server) is the maximal number of bits sent from the user to any single server, and the* answer complexity *(per server) is the maximal number of answer bits sent by any server.*

[1] All the protocols constructed in this paper, as well as all previous information-theoretic PIR protocols, require a single round of queries and answers. This definition may be extended to multi-round PIR in a natural way.

[2] For this definition it is enough to consider collusions of exactly t servers (unlike private computations where parties have different inputs).

2.3 Secret-Sharing

Threshold secret-sharing schemes [18, 56] are an important tool in the construction of several PIR protocols. See [8] for a discussion on the usage of secret sharing in PIR protocols. Informally, a t-out-of-ℓ secret sharing scheme enables a user to share a given secret amongst ℓ users such that only subsets of at least t users can reconstruct the secret, and any subset of less than t users gets no information on the secret. We next describe Shamir's secret sharing scheme [56] which we use in our protocols.

2.3.1 Shamir's scheme [56]

Let $\mathbb{F}$ be a finite field with $q > \ell$ elements, and let $\omega_1, \ldots, \omega_\ell$ be distinct nonzero elements of $\mathbb{F}$. In order to share a secret $s \in \mathbb{F}$ using Shamir's t-out-of-ℓ secret sharing scheme, the dealer chooses $t-1$ random elements $a_{t-1}, \ldots, a_1$, which together with the secret s define a univariate polynomial $p(Y) \stackrel{\text{def}}{=} a_{t-1}Y^{t-1} + a_{t-2}Y^{t-2} + \cdots + a_1 Y + s$. Observe that $p(0) = s$. The share of the j-th player is $p(\omega_j)$. Each set of at least t players can recover $p(Y)$ by interpolation, and hence can also reconstruct $s = p(0)$. More formally, for every set $\{j_1, \ldots, j_t\}$ there exist constants $\alpha_{j_1}, \ldots, \alpha_{j_t}$ (independent of $p(Y)$ and s) where $\alpha_{j_h} \stackrel{\text{def}}{=} \prod_{d \neq h} \frac{\omega_{j_d}}{\omega_{j_d} - \omega_{j_h}}$ such that $s = p(0) = \sum_{h=1}^{t} \alpha_{j_h} p(\omega_{j_h})$. On the other hand, every set of $t-1$ players learns nothing on s from their shares.

In the previous scheme we shared one element of the field; we extend this notion in the natural way to a scheme for sharing of a vector of elements in the field. Given a vector $\mathbf{V} = \langle s^1, \ldots, s^m \rangle$ of length m, i.e., $\mathbf{V} \in \mathbb{F}^m$, we define the shares of the vector, denoted by $\langle \mathbf{V_1}, \ldots, \mathbf{V}_\ell \rangle$, where each $\mathbf{V_j}$ is a vector in $\mathbb{F}^m$ as follows: For each element s^a, where $1 \leq a \leq m$, the user executes Shamir's t-out-of-ℓ secret sharing scheme independently over the field $\mathbb{F}$ producing ℓ shares $s^a_1, \ldots, s^a_\ell$. We then define the vector $\mathbf{V_j}$ as $\langle s^1_j, \ldots, s^m_j \rangle$, i.e., the j-th share out of each set of shares.

3 Generic Transformations

In this section we present several generic transformations from PIR protocols to robust PIR protocols. We start with a warmup transformation from 2-out-of-2 PIR protocols to robust 2-out-of-ℓ PIR protocols. We then generalize this transformation to a transformations from k-out-of-k PIR protocols to robust k-out-of-ℓ PIR protocols and from t-private k-out-of-k PIR protocols to robust t-private k-out-of-ℓ PIR protocols. Finally, we present a more generalized reduction from k-out-of-k PIR protocols to k-out-of-ℓ PIR protocols. The last transformation does not lead to better k-out-of-ℓ PIR protocols when applied to current k-out-of-k PIR protocols. However, if there would be better PIR protocols then this transformation can lead to better protocols.

3.1 A Replication Solution for Robust 2-out-of-ℓ PIR

In this section we will construct a generic transformation from 2-out-of-2 PIR protocols to 2-out-of-ℓ PIR protocols, proving the next theorem:

Theorem 3.1 *If there is a 2-out-of-2 PIR protocol with communication* $\mathrm{PIR}_2(n)$ *per server, then there is a 2-out-of-ℓ PIR protocol with communication* $\mathrm{PIR}_2(n)\log\ell$ *per server and total communication* $\mathrm{PIR}_2(n)\ell\log\ell$.

Proof: Let $\mathcal{P}$ be a 2-out-of-2 PIR protocol. Given a retrieval index i, the user executes the given PIR protocol $\mathcal{P}$ to produce $\log\ell$ independent pairs of queries $\{Q_1, \ldots, Q_{\log\ell}\}$ for the retrieval of x_i, each pair comprising of 2 queries, i.e., $Q_j = \langle Q_j[0], Q_j[1]\rangle$ where $Q_j[a]$ is the query for server a. Each server $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$ receives one query out of each pair of queries and answers this query. (We will describe the algorithm that chooses one query out of each pair later.) The queries sent to each server guarantee that, if the user receives correct answers from at least 2 servers, then there exists an index m such that the user receives an answer for query $Q_m[0]$ and $Q_m[1]$, and thus can reconstruct the bit x_i. (This is done independently of the answers that the user receives or does not receive for the other queries).

We next explain which queries each server receives. Given a server $\mathcal{S}_j$, we look at the representation $b_1^j b_2^j \ldots b_{\log \ell}^j$ of j as a binary number of length $\log \ell$. The user sends the following $\log \ell$ queries to $\mathcal{S}_j$ – for each $1 \leq a \leq \log \ell$ send the query $Q_a[b_a^j]$ to $\mathcal{S}_j$, i.e., if $b_a^j = 0$ send $Q_a[0]$ and if $b_a^j = 1$ send $Q_a[1]$. Each server, upon receiving the queries, replies independently to each query according to the PIR protocol. That is, Server $\mathcal{S}_j$ replies to each query $q = Q_a[b_a^j]$ as Server $\mathcal{S}_{b_a^j}$ would reply to q in the original 2-out-of-2 PIR protocol $\mathcal{P}$. Since we assume that at least 2 servers are reachable, the user receives answers from at least 2 servers, say server $\mathcal{S}_{j_1}$ and server $\mathcal{S}_{j_2}$ (where $j_1 \neq j_2$). The binary representations of j_1 and j_2 differ in at least one bit; let a be the index of the first bit that differs between j_1 and j_2, and without loss of generality, $b_a^{j_1} = 0$ and $b_a^{j_2} = 1$. The user takes the answer received from server $\mathcal{S}_{j_1}$ for query $Q_a[0]$ and the answer received from server $\mathcal{S}_{j_2}$ for query $Q_a[1]$ and reconstructs the desired bit x_i.

Security. This scheme is secure since each server receives only one query out of each pair of queries and these pairs of queries are independent.

Communication Complexity. In this protocol each server receives $\log \ell$ queries and answers each of them, so that the complexity of the protocol is the number of queries multiplied by $\mathrm{PIR}_2(n)$, i.e., the communication per server is $O(\mathrm{PIR}_2(n) \log \ell)$ and the total communication is $O(\mathrm{PIR}_2(n)\ell \log \ell)$. □

Plugging the PIR protocol of [23] we get:

Corollary 3.2 *There exists a 2-out-of-ℓ PIR protocol with total communication of $O(n^{\frac{1}{3}} \ell \log \ell)$.*

3.2 A Generic k-out-of-ℓ Replication Solution

In this section we will generalize the solution presented in the previous section, and construct a generic transformation from k-out-of-k PIR protocols to k-out-of-ℓ PIR protocols. The idea is similar to the 2-out-of-ℓ PIR protocol; however, we need to be more careful in partitioning the queries. For this purpose we recall the following definition:

Definition 3.3 (Perfect Hashing and Minimal Perfect Hashing) *Let $k \leq m \leq \ell$. An (ℓ, k, m)* perfect hash family *$\{h_1, \ldots, h_w\}$ is a family of functions of the form: $h_a : [\ell] \to [m]$ such that for each subset $A \subseteq [\ell]$, where $|A| = k$, there exists an index a such that $|h_a(A)| = k$ (that is, h_a restricted to A is one-to-one). The* size *of the family is the number of functions in the family denoted by w. An (ℓ, k)* minimal perfect hash family *is an (ℓ, k, m) perfect hash family.*

We have 4 parameters for a perfect hash family: ℓ – size of the domain, k – size of hashed sets, m – size of the range, and w – number of functions in the perfect hash family. The size of the range m has to be at least k, since we require that there exists a function that is one-to-one when restricted to a set of size k. Thus, a minimal perfect hash family has the smallest size of range possible for a given k. The parameters ℓ,

m, and k are part of the specification of the problem. We would like the size of the family w to be as small as possible, since w will directly affect our protocol's complexity.

Theorem 3.4 *If there exists an* (ℓ, k) *minimal perfect hash family of size* $w_{\ell,k}$ *and a* k*-out-of-*k *PIR protocol with communication* $\mathrm{PIR}_k(n)$ *per server, then there exists a* k*-out-of-*ℓ *PIR protocol with communication* $w_{\ell,k} \cdot \mathrm{PIR}_k(n)$ *per server, and thus with total communication* $\ell \cdot w_{\ell,k} \cdot \mathrm{PIR}_k(n)$.

Proof: Given a k-out-of-k PIR protocol $\mathcal{P}$ we do the following. Given i, the retrieval index, the user uses $\mathcal{P}$ to produce $w_{\ell,k}$ independent vectors of queries $\left\{Q_1, \ldots, Q_{w_{\ell,k}}\right\}$ for the retrieval of x_i, each vector comprising of k queries, i.e., $Q_j = \langle Q_j[1], \ldots, Q_j[k]\rangle$, that is, the user executes $w_{\ell,k}$ times the protocol $\mathcal{P}$ independently and generates $w_{\ell,k}$ query vectors. Each server receives from the user one query out of each vector of queries and answers this query. Since each server receives $w_{\ell,k}$ PIR queries, which are independent, the server gains no knowledge on i. We show below how the user chooses which queries to send to each server. This choice of queries sent to each server guarantees that if the user receives answers from at least k servers, then it can reconstruct x_i.

Given an (ℓ, k) minimal perfect hash family $\mathcal{H}_{\ell,k} = \{h_a : a \in [w_{\ell,k}]\}$, for every $j \in [k]$, the user sends the following $w_{\ell,k}$ queries to $\mathcal{S}_j$: For each $1 \le a \le w_{\ell,k}$, let $\Delta = h_a(j)$. The user sends $Q_a[\Delta]$ to $\mathcal{S}_j$, i.e., the user sends the Δ-th query out of the vector Q_a. Formally, the query sent to $\mathcal{S}_j$ is $\langle Q_a[h_a(j)]\rangle_{1\le a\le w_{\ell,k}}$. In other words, the minimal perfect hash family determines which queries we need to take from each vector of queries Q_a. Server $\mathcal{S}_j$ replies to each query $q = Q_a[h_a(j)]$, where $1 \le a \le w_{\ell,k}$, as Server $\mathcal{S}_{h_a(j)}$ would reply to q in the original k-out-of-k PIR protocol $\mathcal{P}$.

Let $\mathcal{S}_{j_1}, \ldots, \mathcal{S}_{j_k}$ be k servers from which the user receives answers. By the definition of perfect hashing, there is an index a such that $|h_a(\{j_1, \ldots, j_k\})| = k$, i.e., the set $\{h_a(j_1), \ldots, h_a(j_k)\}$ is exactly $[k]$. We consider the answers received from these servers to the queries $\{Q_a[h_a(j_1)], \ldots, Q_a[h_a(j_k)]\}$. Since these queries are distinct, we have k answers in a k-out-of-k PIR protocol, and the user can reconstruct x_i from the answers received for these queries. $\square$

The communication complexity of the above protocol depends on the size of the minimal perfect hash family. Mehlhorn [48] proved that there exists an (ℓ, k) minimal hash family of size $\log \ell \cdot 2^{O(k)}$ (this is basically optimal [48]). Combining constructions of [58, 48], we get an explicit minimal hash family of the same size as stated in the next claim. See Appendix A for the proof.

Claim 3.5 ([58, 48]) *For every integers* ℓ *and* k*, there is an explicit* (ℓ, k) *minimal perfect hash family of size* $\log \ell \cdot 2^{O(k)}$.

Using the protocol of [3, 38, 37, 8, 65] and the construction of hash family of Claim 3.5 we get:

Corollary 3.6 *There is a* k*-out-of-*ℓ *PIR protocol with total communication* $2^{O(k)} n^{\frac{1}{2k-1}} \ell \log \ell$.

Applying the protocol of [11] and the hash family Claim 3.5 we get:

Corollary 3.7 *For every* $k \ge 3$*, there is a* k*-out-of-*ℓ *PIR protocol with total communication*

$$2^{\tilde{O}(k)} n^{\frac{2 \log\log k}{k \log k}} \ell \log \ell.$$

We use the same approach taken in Theorem 3.4, only this time instead of using a "regular" k-out-of-k PIR protocol, we use a t-private k-out-of-k PIR protocol to produce the $w_{\ell,k}$ independent query vectors. The other details are the same as in the previous transformation.

Theorem 3.8 *If there is an (ℓ, k) minimal perfect hash family of size $w_{\ell,k}$ and a t-private k-out-of-k PIR protocol with communication $\mathrm{PIR}_{k,t}(n)$ per server, then there is a t-private k-out-of-ℓ PIR protocol with communication $w_{\ell,k} \cdot \mathrm{PIR}_{k,t}(n)$ per server, thus total communication $\ell \cdot w_{\ell,k} \cdot \mathrm{PIR}_{k,t}(n)$.*

Applying the protocol of [8] and the hash family of Claim 3.5 we get:

Corollary 3.9 *There is a t-private k-out-of-ℓ PIR protocol with total communication*

$$O(2^{O(k)} \cdot n^{1/\lfloor(2k-1)/t\rfloor} \ell \log \ell).$$

3.3 A Generalized Transformation

We now show a generalization of the previous transformation, where our goal is to reduce the dependency on k. Thus, we first generalize the notion of perfect hashing.

Definition 3.10 (Nearly Perfect Hashing) *An (ℓ, k, α) nearly perfect hash family $\{h_1, \ldots, h_w\}$ (where $\alpha \leq 1$) is a family of functions $h_a : [\ell] \to [\lfloor \alpha k \rfloor]$ such that for each subset $A \subseteq [\ell]$, where $|A| = k$, there exists an index a such that $|h_a(A)| = \lfloor \alpha k \rfloor$ (that is, the function h_a when restricted to A is on $\lfloor \alpha k \rfloor$).*

Note that when $\alpha = 1$ we get the standard definition of a minimal perfect hash family. We now show how to use (ℓ, k, α) nearly perfect hash families in the construction of k-out-of-ℓ PIR protocols.

Theorem 3.11 *If there is an (ℓ, k, α) nearly perfect hash family of size $w_{\ell,k,\alpha}$ and if there is an $\lfloor \alpha k \rfloor$-out-of-$\lfloor \alpha k \rfloor$ PIR protocol (where $\alpha \leq 1$) with communication $\mathrm{PIR}_{\lfloor \alpha k \rfloor}(n)$ per server, then there exists a k-out-of-ℓ PIR protocol with communication $w_{\ell,k,\alpha} \cdot \mathrm{PIR}_{\lfloor \alpha k \rfloor}(n)$ per server, thus total communication $\ell \cdot w_{\ell,k,\alpha} \cdot \mathrm{PIR}_{\lfloor \alpha k \rfloor}(n)$.*

Proof: This proof is similar to that of Theorem 3.4, only this time we use an $\lfloor \alpha k \rfloor$-out-of-$\lfloor \alpha k \rfloor$ PIR protocol and an α perfect hash family. In order to prove the correctness of this protocol we use the property of the (ℓ, k, α) nearly perfect hash family: Let $\mathcal{S}_{j_1}, \ldots, \mathcal{S}_{j_k}$ be k servers from which the user receives answers. Using the (ℓ, k, α) nearly perfect hash family property, let a be an index such that $|h_a(\{j_1, \ldots, j_k\})| = \lfloor \alpha k \rfloor$. This means that the user has $\lfloor \alpha k \rfloor$ answers of an $\lfloor \alpha k \rfloor$-out-of-$\lfloor \alpha k \rfloor$ PIR protocol, and the user can reconstruct x_i from the answers received for these queries. $\Box$

In the last proof we used, as our building block to construct a k-out-of-ℓ PIR protocol, an $\lfloor \alpha k \rfloor$-out-of-$\lfloor \alpha k \rfloor$ PIR protocol (as opposed to Theorem 3.4 where we used a k-out-of-k PIR protocol). Since the communication complexity of PIR protocols decreases as k gets bigger and since we are using $\alpha < 1$, we will get a less efficient PIR protocol in its dependency on n; our hope is that $w_{\ell,k,\alpha}$ is considerably smaller thus the dependency on k will be better. For $\alpha = 1/\ln k$ we show by a standard probabilistic construction that there is a family whose size is small.

Claim 3.12 *There exists an $(\ell, k, \frac{1}{\ln k})$ nearly perfect hash family of size $O(\frac{k \log \ell}{\log \log k})$, where $k \geq 3$.*

Proof: We will prove the claim using a probabilistic proof. As a first step let us consider a specific subset $A \subseteq \{1, \ldots, \ell\}$, where $|A| = k$, one hash function h chosen at random from the space of functions from $\{1, \ldots, \ell\}$ to $\{1, \ldots, \lfloor \alpha k \rfloor\}$ and one index $c \in \{1, \ldots, \lfloor \alpha k \rfloor\}$. Consider the probability

$$\Pr\left[\forall_{j \in A} h(j) \neq c\right] = \left(\frac{\lfloor \alpha k \rfloor - 1}{\lfloor \alpha k \rfloor}\right)^k \leq \left(\left(1 - \frac{1}{\alpha k}\right)^{\alpha k}\right)^{\frac{1}{\alpha}} < e^{-1/\alpha} = \frac{1}{k}.$$

The last equality is true since $\alpha = \frac{1}{\ln k}$. By the union bound we conclude that

$$\Pr[\,|h(A)| < \lfloor \alpha k \rfloor\,] = \Pr[\,\exists_c \forall_{j \in A}\, h(j) \neq c\,] < \lfloor \alpha k \rfloor \frac{1}{k} \leq \alpha = \frac{1}{\ln k}. \tag{1}$$

As the next step we choose $w_{\ell,k,\alpha}$ hash functions independently from the space of functions from $\{1,\ldots,\ell\}$ to $\{1,\ldots,\lfloor \alpha k \rfloor\}$. Thus for a fixed set A we get

$$\Pr\left[\forall_{1 \leq a \leq w_{\ell,k,\alpha}}\, |h_a(A)| < \lfloor \alpha k \rfloor\,\right] < \left(\frac{1}{\ln k}\right)^{w_{\ell,k,\alpha}}.$$

Therefore,

$$\Pr\left[\,\exists_{A;\,|A|=k}\, \forall_{1 \leq a \leq w_{\ell,k,\alpha}}\, |h_a(A)| < \lfloor \alpha k \rfloor\,\right] < \binom{\ell}{k}\left(\frac{1}{\ln k}\right)^{w_{\ell,k,\alpha}} \leq \ell^k \left(\frac{1}{\ln k}\right)^{w_{\ell,k,\alpha}}.$$

If $\ell^k(\frac{1}{\ln k})^{w_{\ell,k,\alpha}} < 1$, then choosing at random $w_{\ell,k,\alpha}$ hash functions, the probability that this family of hash functions is not an (ℓ,k,α) nearly perfect hash family is smaller than 1, i.e., there exists an (ℓ,k,α) nearly perfect hash family of size $w_{\ell,k,\alpha}$. Thus, it suffices that $\ell^k < (\log k)^{w_{\ell,k,\alpha}}$, i.e., $w_{\ell,k,\alpha} > \frac{k \log \ell}{\log \log k}$. $\square$

In the above analysis, (1) could have been derived from the so-called coupon collector problem, see, e.g., [49, pages 57–63]. The analysis of the coupon collector problem implies that if we try to take $\alpha \geq \frac{2}{\ln k}$ then for a given A of size k the probability that $|h(A)| = \lfloor \alpha k \rfloor$ would be exponentially small, thus the size of family we would construct using the above proof would be exponential in k.

With the current state of the art of PIR protocols we cannot achieve a more efficient robust PIR protocols using the $(\ell, k, 1/\ln k)$ nearly perfect hash family. If, for example, there exists a PIR protocol with communication $\text{poly}(k) \cdot n^{O(\frac{1}{k \log k})}$ then we will get a robust protocol with communication complexity of $\text{poly}(k,\ell) \cdot n^{\frac{1}{2k}}$. Notice that the recent PIR protocols [11] are close to these requirements (however, they are not polynomial in k).

4 A k-out-of-ℓ Polynomial Interpolation based PIR Protocol

In this section we construct a k-out-of-ℓ PIR protocol using the polynomial interpolation based PIR protocol of [5, 6, 23]. We start with a known technical lemma (and supply its proof for completeness), and then present the protocol.

Lemma 4.1 *Let d and m be integers such that $m \geq d \cdot n^{\frac{1}{d}}$. There is a function $E : \{1,\ldots,n\} \to \{0,1\}^m$ and an m-variate degree d polynomial P_x (which depends on the database x) such that $P_x(E(i)) = x_i$ for each $1 \leq i \leq n$.*

Proof: Let $E(1),\ldots,E(n)$ be n distinct binary vectors of length m and weight d (such vectors exist since $\binom{m}{d} \geq (m/d)^d \geq n$), let $E(i)_a$ be the a-th bit of $E(i)$ and define

$$P_x(Z_1,\ldots,Z_m) \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i \prod_{a:E(i)_a=1} Z_a.$$

$\square$

Lemma 4.2 *There exists a k-out-of-ℓ PIR protocol with query complexity $O(kn^{\frac{1}{k-1}}\log \ell)$ and answer complexity $O(\log \ell)$ per server.*

Proof: Let $d = k-1$ and P_x and E be as promised in Lemma 4.1. Given a retrieval index i, the user does the following: Calculates the vector $E(i) = \langle y_1, \ldots y_m \rangle$, where y_j is the j-th bit of $E(i)$. Now the user uses Shamir's 2-out-of-ℓ scheme, over a finite field with at least $\ell+1$ elements, to share $E(i)$. That is, it chooses at random m polynomials $\{p_1, \ldots, p_m\}$ (each of degree 1) such that $p_a(0) = y_a$ for each $1 \le a \le m$. Let $\omega_1, \ldots, \omega_\ell$ be distinct nonzero elements of the field. The user sends to server $\mathcal{S}_j$ the shares $\langle p_1(\omega_j), \ldots, p_m(\omega_j)\rangle$.

We now consider the univariate polynomial: $R(Y) = P_x(p_1(Y), \ldots, p_m(Y))$; its degree is d since R is constructed from the polynomial P_x, whose degree is d, by replacing each variable Y_j with a degree 1 polynomial. Given these definitions,

$$R(0) = P_x(p_1(0), \ldots, p_m(0)) = P_x(y_1, \ldots, y_m) = x_i.$$

Furthermore, the server $\mathcal{S}_j$ can compute $R(\omega_j)$ without knowing the polynomial R since

$$R(\omega_j) = P_x(p_1(\omega_j), \ldots, p_m(\omega_j))$$

and since $\langle p_1(\omega_j), \ldots, p_m(\omega_j)\rangle$ are the shares that server $\mathcal{S}_j$ receives from the user. Thus, $\mathcal{S}_j$ computes $R(\omega_j)$ and sends it to the user.

Upon receiving any k answers $R(\omega_{j_1}), \ldots, R(\omega_{j_k})$ (from k different servers) the user reconstructs the polynomial R by interpolation (since the user has k points on a polynomial of degree $d = k-1$) and computes $R(0) = x_i$.

Server $\mathcal{S}_j$ does not gain any information on i since $\mathcal{S}_j$ receives one share of each secret bit $y_1, \ldots, y_m$ in a 2-out-of-ℓ secret sharing scheme. Thus, the protocol is private. The user sends m shares to each server, each share of size $O(\log \ell)$. Each server sends an answer of length $O(\log \ell)$ and thus the total communication is $O(m\ell \log \ell) = O(kn^{\frac{1}{k-1}}\ell \log \ell)$. □

In the above protocol, the answer complexity is smaller than the query complexity. We will balance these complexities using the balancing technique of [23], yielding the following theorem:

Theorem 4.3 *There exists a k-out-of-ℓ PIR protocol with total communication $O(kn^{\frac{1}{k}}\ell \log \ell)$.*

Proof: This communication complexity is achieved by balancing the complexities of the queries and answers of the protocol described in Lemma 4.2, i.e., reducing the query complexity and increasing the answer complexity. This is done by looking at the database as a matrix of size $\alpha(n) \times \frac{n}{\alpha(n)}$, where $\alpha(n)$ will be determined later. We consider each index i as a cell (i_1, i_2), where $i \to (i_1, i_2)$ in a natural mapping of i according to the size of the matrix. To achieve the balancing, the user executes the above PIR protocol with retrieval index i_2 and database of size $\alpha(n)$. Each server considers each row of the matrix as a database of size $\alpha(n)$, and sends the answer to the query it gets for each row. The user then takes the answers it gets for row i_1 and reconstructs x_{i_1,i_2}. The user sends one query to each server, thus the query complexity is $O(\log \ell \cdot k\alpha(n)^{\frac{1}{k-1}})$ per server. Each server sends one answer per row, each answer being of length $O(\log \ell)$; thus, the answer complexity is $\log \ell \cdot \frac{n}{\alpha(n)}$ per server. To minimize the total communication complexity we require: $\log \ell \cdot \frac{n}{\alpha(n)} = O(\log \ell \cdot k\alpha(n)^{\frac{1}{k-1}})$. Taking $\alpha(n) = O(n^{\frac{k-1}{k}})$, we obtain a protocol with total communication $O(kn^{\frac{1}{k}}\ell \log \ell)$. □

A similar construction works for t-private robust protocols.

Lemma 4.4 *There exists a t-private k-out-of-ℓ PIR protocol with query complexity*

$$O\left(\frac{k}{t}n^{1/\lfloor (k-1)/t\rfloor}\log \ell\right)$$

and answer complexity of $O(\log \ell)$ per server.

Proof: Let $d = \lfloor \frac{k-1}{t} \rfloor$, $m = \Theta(dn^{\frac{1}{d}})$, and P_x and E be as promised in Lemma 4.1. The protocol we construct is similar to the protocol described in the proof of Lemma 4.2 with the following differences: In the t-private protocol the user uses Shamir's $(t+1)$-out-of-ℓ scheme, that is, the degree of the polynomials $p_1, \ldots, p_m$ is t. Thus, the degree of R is $dt = \lfloor \frac{k-1}{t} \rfloor \cdot t \leq k-1$. The user, upon receiving any k answers $R(\omega_{j_1}), \ldots, R(\omega_{j_k})$ (from k different servers), reconstructs the polynomial R by interpolation (since the user has k points on a polynomial of degree $k-1$) and computes $R(0) = x_i$.

Security and t-Privacy. A coalition of t servers does not gain any information on i, since we used Shamir's $(t+1)$-out-of-ℓ secret sharing scheme. Thus, the protocol is t-private.

Communication Complexity. The user sends m shares to each server, each share is of size $O(\log \ell)$. Each server sends an answer of length $O(\log \ell)$ and thus the total communication is

$$O(m\ell \log \ell) = O(\frac{k}{t}n^{\frac{1}{\lfloor (k-1)/t\rfloor}}\ell \log \ell).$$

□

Again we apply the balancing technique:

Theorem 4.5 *There is a t-private k-out-of-ℓ PIR protocol with total communication*

$$O\left(\frac{k}{t}n^{\frac{1}{\lfloor (k-1)/t\rfloor+1}}\ell \log \ell\right) = O\left(\frac{k}{t}n^{t/k}\ell \log \ell\right).$$

Proof: We use here the technique used in proof of Theorem 4.3. The query complexity is $O(\frac{k}{t}\log \ell \cdot \alpha(n)^{\frac{1}{\lfloor (k-1)/t\rfloor}})$ per server. Each server sends $n/\alpha(n)$ answers each of length $\log \ell$, thus in order to minimize the total communication complexity we require:

$$\log \ell \cdot \frac{n}{\alpha(n)} = O\left(\frac{k}{t}\log \ell \cdot \alpha(n)^{\frac{1}{\lfloor (k-1)/t\rfloor}}\right).$$

Taking $\alpha(n) = O(n^{\lfloor \frac{k-1}{t} \rfloor / \lfloor \frac{k-1}{t}+1 \rfloor})$, yields a protocol with the desired communication complexity. □

5 A Robust PIR Protocol Using Shamir's Secret Sharing

In this section we show how one can use Shamir's secret sharing to produce robust PIR protocols. We first construct a k-out-of-ℓ PIR protocol with total communication complexity of $O(n\ell \log \ell)$. This protocol is just a "warmup" (because the result is trivial). However, the ideas of this protocol are used to construct a 2-out-of-ℓ protocol whose complexity is $O(n^{1/3}\ell \log \ell)$.

Given the retrieval index i, the user shares the unit vector $\mathbf{e_i}$ of length n using Shamir's k-out-of-ℓ secret sharing scheme (as described in Section 2.3.1) over GF($2^{\lfloor \log \ell \rfloor + 1}$). Denote by $\langle \mathbf{V_1}, \ldots, \mathbf{V}_\ell \rangle$ the shares the user computed. The user sends $\mathbf{V_j}$ to server $\mathcal{S}_j$ for each $1 \leq j \leq \ell$. Server $\mathcal{S}_j$, upon receiving $\mathbf{V_j}$, sends back to the user the following scalar product: $a_j \stackrel{\text{def}}{=} \mathbf{V_j} \cdot \mathbf{x}$, i.e., the server computes the scalar product of the database and the vector $\mathbf{V_j}$ and sends the result to the user.

The user upon receiving k answers $a_{j_1}, \ldots, a_{j_k}$ uses the appropriate constants $\alpha_{j_1}, \ldots, \alpha_{j_k}$ (from Section 2.3.1) to perform the following computation (in the following proof all additions and multiplications are done in GF($2^{\lfloor \log \ell \rfloor + 1}$)):

$$\sum_{h=1}^{k} \alpha_{j_h} a_{j_h} = \sum_{h=1}^{k} \alpha_{j_h} (\mathbf{V_{j_h}} \cdot \mathbf{x}) = (\sum_{h=1}^{k} \alpha_{j_h} \mathbf{V_{j_h}}) \cdot \mathbf{x} = \mathbf{e_i} \cdot \mathbf{x} = x_i.$$

Thus, the user can reconstruct x_i from any k answers.

Security. This protocol is secure since the user sends each server one share of the vector $\mathbf{e_i}$. Since Shamir's scheme is secure the server cannot gain any information about i from the shares it received.

Communication Complexity. The user sends ℓ vectors, each of length $n \log \ell$, and each server sends an answer of length $\log \ell$, and thus we get communication complexity of $O(n\ell \log \ell)$. Using the balancing technique of [23] we can reduce the total communication complexity to $O(n^{1/2} \ell \log \ell)$.

We now present a more efficient protocol that uses the above ideas combined with the 2-server protocol of [23]. This 2-out-of-ℓ protocol works with total communication of $O(n^{1/3} \ell \log \ell)$. Let us first recall the protocols presented by [23]:

ORIGINAL PROTOCOL (VARIANT OF [23]). Let $n = m^3$ for some m, and consider the database as a 3-dimensional cube, i.e., every $i \in [n]$ is represented as $\langle i_1, i_2, i_3 \rangle$ where $i_r \in [m]$ for $r = 1, 2, 3$. This is done using a natural mapping from $[n]$ to $[m]^3$. Furthermore, we use the following notation.

Notation 5.1 *Let x be a three-dimensional cube of height m. We denote by $x_{j_1,*,*}$ the matrix of all values of x where the first index of this value is j_1. Formally, we define $x_{j_1,*,*}$ as the matrix A where $A_{i_1,i_2} = x_{j_1,i_1,i_2}$. We define $x_{*,j_1,*}$ and $x_{*,*,j_1}$ similarly. We denote by $x_{j_1,j_2,*}$ the vector obtained from the 3-dimensional cube x by taking all the values of x where the first index of this value is j_1 and the second is j_2. Formally, we define $x_{j_1,j_2,*}$ as the vector $\mathbf{A}$ where $A[i_1] = x_{j_1,j_2,i_1}$. We define x_{*,j_1,j_2} and $x_{j_1,*,j_2}$ similarly.*

In Figure 1 we describe the protocol. It can be checked that each bit, except for x_{i_1,i_2,i_3}, appears an even number of times in the exclusive-or the user computes in Step 3, and thus cancels itself. Therefore, the user outputs x_{i_1,i_2,i_3} as required. Furthermore, the communication is $O(m) = O(n^{1/3})$.

We may consider $\mathbf{A_r^2} = \mathbf{A_r^1} \oplus \mathbf{e_{i_r}}$ and $\mathbf{A_r^1}$ as two shares in a 2-out-of-2 sharing scheme of the unit vector $\mathbf{e_{i_r}}$. We use a similar approach to construct a robust protocol. There is one difference – we will use Shamir's 2-out-of-ℓ secret sharing scheme in order to share the unit vector $\mathbf{e_{i_r}}$; these shares are used to generate the queries for the protocol.

Theorem 5.2 *There exists a 2-out-of-ℓ PIR protocol with total communication of $O(n^{1/3} \ell \log \ell)$.*

Proof: In this proof we consider the database x as a 3-dimensional cube and use Shamir's 2-out-of-ℓ secret sharing scheme to construct our queries:

The Two-Server Protocol of [23]

1. The user selects three random vectors $\mathbf{A^1_1}, \mathbf{A^1_2}, \mathbf{A^1_3} \in \{0,1\}^m$, and computes $\mathbf{A^2_r} = \mathbf{A^1_r} \oplus \mathbf{e_{i_r}}$ for $r = 1, 2, 3$.

   The user sends $\mathbf{A^j_1}, \mathbf{A^j_2}, \mathbf{A^j_3}$ to $\mathcal{S}_j$ for $j = 1, 2$.

2. Server $\mathcal{S}_j$ computes for every $b \in [m]$

$$\begin{aligned} a^j_{1,b} &\stackrel{\text{def}}{=} \mathbf{A^j_2} \cdot x_{b,*,*} \cdot \mathbf{A^j_3}, \\ a^j_{2,b} &\stackrel{\text{def}}{=} \mathbf{A^j_1} \cdot x_{*,b,*} \cdot \mathbf{A^j_3}, \\ a^j_{3,b} &\stackrel{\text{def}}{=} \mathbf{A^j_1} \cdot x_{*,*,b} \cdot \mathbf{A^j_2}, \end{aligned}$$

   and sends the $3m$ bits $\left\{ a^j_{r,b} \ : \ r \in \{1,2,3\} \ , \ b \in [m] \right\}$ to the user.

3. The user outputs $\bigoplus_{r=1,2,3} (a^1_{r,i_r} \oplus a^2_{r,i_r})$.

Figure 1: The two server protocol of [23] with communication $O(n^{1/3})$.

Given ℓ and retrieval index $i = \langle i_1, i_2, i_3 \rangle$, the user, for every $r \in \{1,2,3\}$, computes the vector $\langle \mathbf{U^r_1}, \ldots, \mathbf{U^r_\ell} \rangle$ as the shares in a Shamir's 2-out-of-ℓ scheme of the unit vector $\mathbf{e_{i_r}}$. The user sends the query $\mathbf{U^1_j}, \mathbf{U^2_j}, \mathbf{U^3_j}$ to server $\mathcal{S}_j$ for each $1 \leq j \leq \ell$.

Server $\mathcal{S}_j$, upon receiving $\mathbf{U^1_j}, \mathbf{U^2_j}, \mathbf{U^3_j}$, sends back to the user the following $3n^{1/3}$ numbers: For each $1 \leq b \leq n^{1/3}$ the server sends to the user the element $\mathbf{U^2_j} \cdot x_{b,*,*} \cdot \mathbf{U^3_j}$, the element $\mathbf{U^1_j} \cdot x_{*,b,*} \cdot \mathbf{U^3_j}$, and the element $\mathbf{U^1_j} \cdot x_{*,*,b} \cdot \mathbf{U^2_j}$ (where $x_{b,*,*}$, $x_{*,b,*}$, and $x_{*,*,b}$ are the two-dimensional matrices described in Notation 5.1), i.e., each number the server sends is a result of multiplications of a two-dimensional matrix produced from the cube with the vectors sent by the user. As in the regular 2-out-of-2 scheme, the user takes one element out of each set of answers: The user upon receiving answers from 2 servers f and g considers the following 6 numbers:

$$\begin{array}{lll} \mathbf{U^2_f} \cdot x_{i_1,*,*} \cdot \mathbf{U^3_f}, & \mathbf{U^1_f} \cdot x_{*,i_2,*} \cdot \mathbf{U^3_f}, & \mathbf{U^1_f} \cdot x_{*,*,i_3} \cdot \mathbf{U^2_f}, \\ \mathbf{U^2_g} \cdot x_{i_1,*,*} \cdot \mathbf{U^3_g}, & \mathbf{U^1_g} \cdot x_{*,i_2,*} \cdot \mathbf{U^3_g}, & \mathbf{U^1_g} \cdot x_{*,*,i_3} \cdot \mathbf{U^2_g}. \end{array} \tag{2}$$

The following claim is similar to the claim that in the protocol of [23] the user reconstructs the correct bit x_i.

Claim 5.3 *There exists a linear combination of the 6 numbers appearing in (2) that computes the desired bit x_{i_1,i_2,i_3}.*

Proof: In our proof we use the constants from Shamir's scheme α_f and α_g (see Section 2.3.1). These two constants are independent of the answers received from the servers. Denote $\mathbf{V^r} \stackrel{\text{def}}{=} \alpha_f \mathbf{U^r_f}$ and $\mathbf{W^r} \stackrel{\text{def}}{=} \alpha_g \mathbf{U^r_g}$ for $r \in \{1,2,3\}$. Thus,

$$\mathbf{V^r} + \mathbf{W^r} = \mathbf{e_{i_r}} \tag{3}$$

for $r \in \{1,2,3\}$.

Notice that $\mathbf{V^2} \cdot x_{i_1,*,*} \cdot \mathbf{V^3} = (\alpha_f)^2(\mathbf{U_f^2} \cdot x_{i_1,*,*} \cdot \mathbf{U_f^3})$. In our computation we multiply each of the first three numbers by α_f^2 and each of the last three numbers by α_g^2 and consider the following combination:

$$\begin{aligned} S \stackrel{\text{def}}{=} & \mathbf{V^2} \cdot x_{i_1,*,*} \cdot \mathbf{V^3} + \mathbf{V^1} \cdot x_{*,i_2,*} \cdot \mathbf{V^3} + \mathbf{V^1} \cdot x_{*,*,i_3} \cdot \mathbf{V^2} \\ & + \mathbf{W^2} \cdot x_{i_1,*,*} \cdot \mathbf{W^3} + \mathbf{W^1} \cdot x_{*,i_2,*} \cdot \mathbf{W^3} + \mathbf{W^1} \cdot x_{*,*,i_3} \cdot \mathbf{W^2}. \end{aligned} \tag{4}$$

Note that S is a linear combination of the bits of the database x. We next prove that $S = x_{i_1,i_2,i_3}$. The proof is somewhat technical and shows that the coefficient of every bit of x, except for x_{i_1,i_2,i_3}, is zero. First, We use the fact that GF($2^{\lfloor \log \ell \rfloor + 1}$) is of characteristic 2 and we can add the number $\mathbf{W^2} \cdot x_{i_1,*,*} \cdot \mathbf{V^3}$ twice without changing the sum. Thus,

$$\begin{aligned} & \mathbf{V^2} \cdot x_{i_1,*,*} \cdot \mathbf{V^3} + \mathbf{W^2} \cdot x_{i_1,*,*} \cdot \mathbf{W^3} && (5) \\ & = \mathbf{V^2} \cdot x_{i_1,*,*} \cdot \mathbf{V^3} + \mathbf{W^2} \cdot x_{i_1,*,*} \cdot \mathbf{V^3} + \mathbf{W^2} \cdot x_{i_1,*,*} \cdot \mathbf{W^3} + \mathbf{W^2} \cdot x_{i_1,*,*} \cdot \mathbf{V^3} && \\ & = (\mathbf{V^2} + \mathbf{W^2}) \cdot x_{i_1,*,*} \cdot \mathbf{V^3} + \mathbf{W^2} \cdot x_{i_1,*,*} \cdot (\mathbf{W^3} + \mathbf{V^3}) && \\ & = \mathbf{e_{i_2}} \cdot x_{i_1,*,*} \cdot \mathbf{V^3} + \mathbf{W^2} \cdot x_{i_1,*,*} \cdot \mathbf{e_{i_3}} && (6) \\ & = x_{i_1,i_2,*} \cdot \mathbf{V^3} + \mathbf{W^2} \cdot x_{i_1,*,i_3}, && (7) \end{aligned}$$

where the equality in (6) follows from (3), and the last equality follows since $\mathbf{e_{i_2}} \cdot x_{i_1,*,*} = x_{i_1,i_2,*}$, and similarly $x_{i_1,*,*} \cdot \mathbf{e_{i_3}} = x_{i_1,*,i_3}$ (multiplication from the right replaces the rightmost $*$ and multiplication from the left replaces the leftmost $*$). That is, already in the sum of the two terms in (5) many bits of x have a zero coefficient. Similarly,

$$\mathbf{V^1} \cdot x_{*,i_2,*} \cdot \mathbf{V^3} + \mathbf{W^1} \cdot x_{*,i_2,*} \cdot \mathbf{W^3} = x_{i_1,i_2,*} \cdot \mathbf{V^3} + \mathbf{W^1} \cdot x_{*,i_2,i_3}, \tag{8}$$

and

$$\mathbf{V^1} \cdot x_{*,*,i_3} \cdot \mathbf{V^2} + \mathbf{W^1} \cdot x_{*,*,i_3} \cdot \mathbf{W^2} = x_{i_1,*,i_3} \cdot \mathbf{V^2} + \mathbf{W^1} \cdot x_{*,i_2,i_3}. \tag{9}$$

Thus, by (4) – (9),

$$\begin{aligned} S &= x_{i_1,i_2,*} \cdot \mathbf{V^3} + \mathbf{W^2} \cdot x_{i_1,*,i_3} + x_{i_1,i_2,*} \cdot \mathbf{W^3} + \mathbf{V^1} \cdot x_{*,i_2,i_3} + x_{i_1,*,i_3} \cdot \mathbf{V^2} + \mathbf{W^1} \cdot x_{*,i_2,i_3} \\ &= x_{i_1,i_2,*} \cdot (\mathbf{V^3} + \mathbf{W^3}) + (\mathbf{V^1} + \mathbf{W^1}) \cdot x_{*,i_2,i_3} + x_{i_1,*,i_3} \cdot (\mathbf{V^2} + \mathbf{W^2}) \\ &= x_{i_1,i_2,*} \cdot \mathbf{e_{i_3}} + \mathbf{e_{i_1}} \cdot x_{*,i_2,i_3} + x_{i_1,*,i_3} \cdot \mathbf{e_{i_2}} \\ &= x_{i_1,i_2,i_3} + x_{i_1,i_2,i_3} + x_{i_1,i_2,i_3} = x_{i_1,i_2,i_3}. \end{aligned}$$

■ (of Claim 5.3)

We now provide the proof of theorem 5.2, by analyzing the privacy and the communication complexity of the protocol appearing in Figure 2. Each server gets one share of Shamir's 2-out-of-ℓ scheme. Since Shamir's scheme is secure, each server cannot gain any information about i from the share it received, and the protocol is secure. Each server sends and receives $3n^{1/3}$ elements of GF($2^{\lfloor \log \ell \rfloor + 1}$), and thus the total communication is $O(n^{1/3}\ell \log \ell)$. □

6 Dealing with Byzantine Servers

In previous sections we assumed that servers may crash. However they cannot reply with wrong answers. We next show solutions for the robust PIR problem tolerating some Byzantine servers, that is, some servers

2-out-of-ℓ PIR protocol which uses Shamir's secret sharing scheme

1. Given ℓ and the retrieval index i_1, i_2, i_3, we denote the vectors $\langle \mathbf{U^r_1}, \dots, \mathbf{U^r_\ell} \rangle$ as the shares of the unit vector $\mathbf{e_{i_r}}$ for $r \in \{1, 2, 3\}$.

2. Server $\mathcal{S}_j$ upon receiving $\mathbf{U^1_j}, \mathbf{U^2_j}, \mathbf{U^3_j}$ sends back to the user the following $3n^{1/3}$ numbers:
For each $1 \leq a \leq n^{1/3}$ the server sends to the user: $\mathbf{U^2_j} \cdot x_{a,*,*} \cdot \mathbf{U^3_j}$, $\mathbf{U^1_j} \cdot x_{*,a,*} \cdot \mathbf{U^3_j}$, and $\mathbf{U^1_j} \cdot x_{*,*,a} \cdot \mathbf{U^2_j}$.

3. The user upon receiving answers from 2 servers $\mathcal{S}_f$ and $\mathcal{S}_g$ computes x_{i_1,i_2,i_3} as the linear combination specified in Claim 5.3:
$$\alpha_f^2(\mathbf{V_f} \cdot x_{i_1,*,*} \cdot \mathbf{W_f} + \mathbf{U_f} \cdot x_{*,i_2,*} + \cdot \mathbf{W_f} + \mathbf{U_f} \cdot x_{*,*,i_3} \cdot \mathbf{V_f}) \\ + \alpha_g^2(\mathbf{V_g} \cdot x_{i_1,*,*} \cdot \mathbf{W_g} + \mathbf{U_g} \cdot x_{*,i_2,*} \cdot \mathbf{W_g} + \mathbf{U_g} \cdot x_{*,*,i_3} \cdot \mathbf{V_g}).$$

Figure 2: A 2-out-of-ℓ PIR protocol which uses Shamir's scheme with total communication $O(n^{1/3} \log \ell)$ per server.

might be malicious or have a corrupted or obsolete database, and may return any answer to the user's queries. The user needs to be prepared for wrong answers from the servers and still reconstruct the right value of the desired bit x_i.

Definition 6.1 (Byzantine-Robust PIR) *A b Byzantine-robust k-out-of-ℓ PIR protocol $\mathcal{P}$ is defined as in Definition 2.1, where the correctness requirement is replaced by the following requirement:*

Correctness. *The user always computes the correct value of x_i from any k answers, of which at least $k-b$ are correct. Formally, for every $i \in \{1, \dots, n\}$, every random string r, every set $K = \{j_1, \dots, j_k\} \subseteq \{1, \dots, \ell\}$, every database $x \in \{0,1\}^n$, and every k answers $\{a_1, \dots, a_k\}$ such that*

$$|\{w \in [k] : a_w = \mathcal{A}(j_w, \mathcal{Q}(j_w, i, r), x)\}| \geq k - b,$$

we have:

$$C(i, r, K, a_1, \dots, a_k) = x_i.$$

We assume that there at least k honest servers available and the user receives at least k answers. However, the system is asynchronous. Thus, upon receiving k answers, the user cannot know if either it received the answers from k honest servers (and possibly no other answers arrive later) or there are some answers from Byzantine servers (and additional answers arrive later).

In Definition 6.1, the correctness holds even if the Byzantine servers cooperate. For the privacy we assume that the Byzantine servers do not cooperate. That is, we deal with 1-privacy (and not, say, b-privacy). Later, we will show what can be done when we discard this assumption. Note that, since we are talking about one-round PIR protocols, the definition is simple. For example, Byzantine servers will not learn any new information as a result of sending wrong answers.

Clearly, if the user receives answers from k servers, then, to enable the user to reconstruct the correct value of x_i, more than half of the answers must be correct. This condition is also sufficient as shown by the following trivial protocol: Each server sends the entire database to the user. Given k answers, out of which less than $k/2$ are Byzantine, the user takes the value of x_i which appears at least $k/2$ times.

Next we show a generic transformation from robust protocols to robust protocols that tolerate Byzantine servers.

Theorem 6.2 *Let a be a parameter where $0 < a \leq k$, and assume there is an a-out-of-ℓ robust PIR protocol with total communication $\mathrm{PIR}_a^\ell(n)$. Then there exists a $\lfloor (k-a)/2 \rfloor$ Byzantine-robust k-out-of-ℓ PIR protocol with total communication $\mathrm{PIR}_a^\ell(n)$.*

Proof: The user and the servers execute a robust a-out-of-ℓ PIR protocol. Assume that the user receives answers from a set B of servers of size at least k. Now, for each subset of size a of B, the user reconstructs x_i (recall that the user can reconstruct x_i from any a answers). The user finds a largest subset $A \subseteq B$ such that for every subset of A of size a the user reconstructs the same value of x_i, and outputs this value as the value of x_i.

We next prove that the user reconstructs the correct value of x_i. Since there are at most $\lfloor (k-a)/2 \rfloor$ Byzantine servers and the size of B is at least k, there are at least $k - \lfloor (k-a)/2 \rfloor = \lceil (k+a)/2 \rceil$ honest servers in B; for every subset of the honest servers of size a the user reconstructs the correct value of x_i. Hence, $|A| \geq \lceil (k+a)/2 \rceil$. Since there are at most $\lfloor (k-a)/2 \rfloor$ Byzantine servers, the set A contains at least $\lceil (k+a)/2 \rceil - \lfloor (k-a)/2 \rfloor \geq a$ honest servers, and therefore the value of x_i reconstructed for this set (and any other subset of A) is the correct value of x_i. □

Plugging the a-out-of-ℓ PIR protocol of Corollaries 3.6 and 3.7 we get:

Corollary 6.3 *For every k and a such that $0 < a \leq k$, there exist $\lfloor (k-a)/2 \rfloor$ Byzantine-robust k-out-of-ℓ PIR protocols with total communication*

1. $2^{O(a)} n^{1/(2a-1)} \ell \log \ell$, *and*
2. $2^{\tilde{O}(a)} n^{\frac{2 \log \log a}{a \log a}} \ell \log \ell$.

In the previous protocol the user is required to reconstruct x_i for $\binom{k}{a}$ sets. We now show a construction which overcomes this exponential dependency on k. We construct a robust k-out-of-ℓ PIR protocol in which (at most) $k/3$ servers are Byzantine. Note that in the following protocol the communication complexity is worse than in the generic protocol.

Theorem 6.4 *There exists a $\lfloor k/3 \rfloor$ Byzantine-robust k-out-of-ℓ PIR protocol with total communication $O(kn^{1/\lfloor k/3 \rfloor} \ell \log \ell)$.*

Proof: We use the $\lfloor k/3 \rfloor$-out-of-ℓ protocols described in the proofs of Lemma 4.2 and Theorem 4.3.[3] In the protocols, the answers of the honest servers are points on a univariate polynomial R whose degree is $\lfloor k/3 \rfloor - 1$. (When we say that the answer a of server $\mathcal{S}_j$ is on R we mean that $a = R(\omega_j)$ where ω_j is defined in Lemma 4.2.) The user needs to interpolate the polynomial R from the answers of the servers. Since some of the servers are Byzantine, not all of the answers are points on R. Nevertheless, we now show

[3]Notice that the balancing technique used in Theorem 4.3 does not change the details.

that the user can still reconstruct R as it is the only polynomial on which at least $\frac{2}{3}$ of the points (answers) reside.

We know that at least $\lceil 2k/3 \rceil$ of the servers are not Byzantine, thus all of these servers send points on R. The user has to find $\lceil 2k/3 \rceil$ points which reside on a polynomial of degree at most $\lfloor k/3 \rfloor - 1$ (i.e., on R) and use this polynomial to reconstruct x_i. By standard arguments, there is exactly one polynomial of degree $\lfloor k/3 \rfloor - 1$ that agrees with at least $\lceil 2k/3 \rceil$ of these k points, and the user can reconstruct this polynomial using the decoding algorithm of the Reed-Solomon error correcting codes [55]. (For more information on error-correcting codes the reader can refer to, e.g., [46].) The communication complexity of the above protocol is the communication complexity of the $\lfloor k/3 \rfloor$-out-of-ℓ protocol of Theorem 4.3 i.e., $O(kn^{1/\lfloor k/3 \rfloor}\ell \log \ell)$. □

Since we consider Byzantine servers, the assumption that they do not cooperate is questionable, thus it might be more reasonable to consider robust b-private PIR protocols in the presence of b Byzantine servers.[4] That is, we consider a robust PIR protocol where the privacy holds even if b Byzantine servers cooperate. We next show two corollaries where we allow the Byzantine servers to cooperate.

Corollary 6.5 *Let a be a parameter where $k/3 < a \leq k$, and define $b = \lfloor (k-a)/2 \rfloor$. Assume there is a robust b-private a-out-of-ℓ PIR protocol with total communication $\mathrm{PIR}^{\ell}_{a,b}(n)$. Then there exists a b-private b Byzantine-robust k-out-of-ℓ PIR protocol with total communication $\mathrm{PIR}^{\ell}_{a,b}(n)$.*

The idea is to use the same approach seen in Theorem 6.2, but with a b-private a-out-of-ℓ PIR protocol. Notice that a b-private a-out-of-ℓ PIR protocol with sub-linear communication exists only if $a > b$, thus we get that $a > k/3$ and $b < k/3$.

Corollary 6.6 *There exists a b-private b Byzantine-robust k-out-of-ℓ PIR protocol (where $b < k/3$) with total communication $O(\frac{k}{b} n^{\frac{1}{\lfloor (k-2)/b \rfloor - 1}} \ell \log \ell)$.*

Proof: We use Corollary 6.5 and Theorem 4.5. To apply Corollary 6.5, we take $b = \lfloor (k-a)/2 \rfloor$ and use a b-private a-out-of-ℓ PIR protocol. This implies that $a \geq k - 2b - 1$, and the communication complexity of the b-private a-out-of-ℓ PIR protocol of Theorem 4.5 is $O(\frac{k}{b} n^{\frac{1}{\lfloor (k-2)/b \rfloor - 1}} \ell \log \ell)$. □

7 Open Problems

We have shown several robust PIR protocols with different features – a protocol with the best known complexity with regard to n, logarithmic in ℓ, but exponential in k, or polynomial in k and ℓ but not optimal in regard to n. Some open questions remain: Given k, ℓ, and n what is the best possible communication complexity for a robust k-out-of-ℓ PIR protocol? I.e., can one present a protocol which is optimal with regard to n and still polynomial in k and ℓ? Given a k-out-of-k PIR protocol, what is the minimum overhead in communication complexity needed in order to transform any protocol to a robust k-out-of-ℓ PIR protocol? That is, can one design a more efficient generic transformation? We note that even the question of the optimal communication complexity of "standard" PIR protocols is a long standing open question.

As demonstrated in this work the robust PIR is a well motivated protocol as is. Furthermore, we believe that this protocol can be used as a building block in more complex protocols. For example, we use it to

[4] We could also consider the more general case of t-private b Byzantine-robust PIR, where possibly $b \neq t$. However, the most interesting case is $b = t$.

construct Byzantine-robust PIR protocols. An open problem is to find other applications and other protocols which use robust PIR protocols.

References

[1] N. Alon. Explicit construction of exponential sized families of k-independent sets. *Discrete Math.*, 58:191–193, 1986.

[2] N. Alon and M. Naor. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16:434–449, 1996.

[3] A. Ambainis. Upper bound on the communication complexity of private information retrieval. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proc. of the 24th International Colloquium on Automata, Languages and Programming*, volume 1256 of *Lecture Notes in Computer Science*, pages 401–407. Springer-Verlag, 1997.

[4] M. Atici, S. S. Magliveras, D. R. Stinson, and W. D. Wei. Some recusrive constructions for perfect hash families. *J. Combin. Des.*, 4:353–363, 1996.

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

[6] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Locally random reductions: Improvements and applications. *J. of Cryptology*, 10(1):17–36, 1997. Early version: Security with small communication overhead, CRYPTO '90, volume 537 of *Lecture Notes in Computer Science*, pages 62-76. Springer-Verlag, 1991.

[7] R. Beigel, L. Fortnow, and W. Gasarch. Nearly tight bounds for private information retrieval systems. *Computational Complexity*, 15(1):82–91, 2006.

[8] A. Beimel and Y. Ishai. Information-theoretic private information retrieval: A unified construction. In P. G. Spirakis and J. van Leeuven, editors, *Proc. of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 912–926. Springer-Verlag, 2001. Journal version in [9].

[9] A. Beimel, Y. Ishai, and E. Kushilevitz. General constructions for information-theoretic private information retrieval. *J. of Computer and System Sciences*, 71(2):213–247, 2005. Journal version of [37] and [8].

[10] A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *Proc. of the 31st ACM Symp. on the Theory of Computing*, pages 89–98, 1999.

[11] A. Beimel, Y. Ishai, E. Kushilevitz, and J. F. Raymond. Breaking the $O(n^{\frac{1}{2k-1}})$ barrier for information-theoretic private information retrieval. In *Proc. of the 43rd IEEE Symp. on Foundations of Computer Science*, pages 261–270, 2002.

[12] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *J. of Cryptology*, 17(2):125–151, 2004. Preliminary version: M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 56–74. Springer, 2000.

[13] A. Beimel and Y. Stahl. Robust information-theoretic private information retrieval. In S. Cimato, C. Galdi, and G. Persiano, editors, *3rd Conf. on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 326–341. Springer-Verlag, 2002.

[14] S. R. Blackburn. Combinatorial designs and their applications. *Research Notes in Mathematics*, 403:44–70, 1999.

[15] S. R. Blackburn. Perfect hash families: Probabilistic methods and explicit constructions. *J. of Combinatorial Theory, Series A*, 92:54–60, 2000.

[16] S. R. Blackburn, M. Burmester, Y. Desmedt, and P. R. Wild. Efficient multiplicative sharing schemes. In U. Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 107–118, 1996.

[17] S. R. Blackburn and P. R. Wild. Optimal linear perfect hash families. *J. of Combinatorial Theory, Series A*, 83:233–250, 1998.

[18] G. R. Blakley. Safeguarding cryptographic keys. In R. E. Merwin, J. T. Zanca, and M. Smith, editors, *Proc. of the 1979 AFIPS National Computer Conference*, volume 48 of *AFIPS Conference proceedings*, pages 313–317. AFIPS Press, 1979.

[19] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer-Verlag, 1999.

[20] R. Canetti, Y. Ishai, R. Kumar, M. K. Reiter, R. Rubinfeld, and R. N. Wright. Selective private function evaluation with applications to private statistics. In *Proc. of the 20th ACM Symp. on Principles of Distributed Computing*, pages 293 – 304, 2001.

[21] Y.-C. Chang. Single database private information retrieval with logarithmic communication. In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004*, volume 3108 of *Lecture Notes in Computer Science*, pages 50 – 61. Springer-Verlag, 2004.

[22] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of the 29th ACM Symp. on the Theory of Computing*, pages 304–313, 1997.

[23] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of the 36th IEEE Symp. on Foundations of Computer Science*, pages 41–51, 1995. Journal version: *J. of the ACM*, 45:965–981, 1998.

[24] Z. J. Czech, G. Havas, and B. S. Majewski. Perfect hashing. *Theoretical Computer Science*, 182:1–143, 1997.

[25] A. Deshpande, R. Jain, T Kavita, V. Lokam, and J. Radhakrishnan. Lower bounds for adaptive locally decodable codes. *Random Structures & Algorithms*, 27(3):358–378, 2005. Conference version: *Proc. of the 17th IEEE Conf. on Computational Complexity*, pages 184–193, 2002.

[26] G. Di-Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for private information retrieval. *J. of Cryptology*, 14(1):37–74, 2001. Preliminary version in *Proc. of the 17th ACM Symp. on Principles of Distributed Computing*, pages 91–100, 1998.

[27] G. Di-Crescenzo, T. Malkin, and R. Ostrovsky. Single-database private information retrieval implies oblivious transfer. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 122–138. Springer-Verlag, 2000.

[28] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright. Secure multiparty computation of approximations. In P. G. Spirakis and J. van Leeuven, editors, *Proc. of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 927–938. Springer-Verlag, 2001.

[29] A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer-Verlag, 1994.

[30] M. L. Fredman and J. Komlós. On the size of separating systems and families of perfect hash funtions. *SIAM Journal on Algebraic and Discrete Methods*, 5:61–68, 1984.

[31] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *J. of the ACM*, 31(3):538–544, 1984.

[32] W. Gasarch. A survey on private information retrieval. *Bulletin of the European Association for Theoretical Computer Science*, 82:72–107, 2004. Also can be found at: http://www.cs.umd.edu/$\sim$gasarch/pir/pir.html.

[33] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Proc. of the 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815. Springer-Verlag, 2005.

[34] Y. Gertner, S. Goldwasser, and T. Malkin. A random server model for private information retrieval. In M. Luby, J. Rolim, and M. Serna, editors, *RANDOM '98, 2nd International Workshop on Randomization and Approximation Techniques in Computer Science*, volume 1518 of *Lecture Notes in Computer Science*, pages 200–217. Springer-Verlag, 1998.

[35] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. of Computer and System Sciences*, 60(3):592–629, 2000.

[36] O. Goldreich, H. Karloff, L. J. Schulman, and L. Trevisan. Lower bounds for linear locally decodable codes and PIR. In *Proc. of the 17th IEEE Conf. on Computational Complexity*, pages 175–183, 2002.

[37] Y. Ishai and E. Kushilevitz. Improved upper bounds on information theoretic private information retrieval. In *Proc. of the 31st ACM Symp. on the Theory of Computing*, pages 79 – 88, 1999. Journal version in [9].

[38] T. Itoh. Efficient private information retrieval. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E82-A(1):11–20, 1999.

[39] T. Itoh. On lower bounds for the communication complexity of private information retrieval. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E84-A(1):157–164, 2001.

[40] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proc. of the 32nd ACM Symp. on the Theory of Computing*, pages 80–86, 2000.

[41] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *J. of Computer and System Sciences*, 69(3):395–420, 2004.

[42] J. Korner and Marton. New bounds for perfect hashing via information theory. *European J. Combin.*, 9:523–530, 1988.

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

[44] E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for single-database computationally-private information retrieval. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 104–121, 2000.

[45] H. Lipmaa. An oblivious transfer protocol with log-squared communication. In J. Zhou and J. Lopez, editors, *the 8th Information Security Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328. Springer-Verlag, 2005.

[46] F. R. Macwilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. Mathematical library. North-Holland, 1978.

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

[48] K. Mehlhorn. *Data structures and Algorithms*, volume 1. Sorting and Searching. Springer-Verlag, 1984.

[49] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[50] M. Naor and B. Pinkas. Oblivious transfer and polynomial evalutation. In *Proc. of the 31st ACM Symp. on the Theory of Computing*, pages 245–254, 1999.

[51] I. Newman and A. Wigderson. Lower bounds on formula size of Boolean functions using hypergraph entropy. *SIAM J. on Discrete Mathematics*, 8:536–542, 1995.

[52] K. Obata. Optimal lower bounds for 2-query locally decodable linear codes. In *RANDOM '02, 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, volume 2483 of *Lecture Notes in Computer Science*, pages 39 – 50. Springer-Verlag, 2002.

[53] R. Ostrovsky and V. Shoup. Private information storage. In *Proc. of the 29th ACM Symp. on the Theory of Computing*, pages 294–303, 1997.

[54] A. Razborov and S. Yekhanin. An $\Omega(n^{1/3})$ lower bound for bilinear group based private information retrrieval. In *Proc. of the 47th IEEE Symp. on Foundations of Computer Science*, 2006.

[55] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. SIAM*, 8:300–304, 1960.

[56] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[57] D. Shiowattana and S. V. Lokam. An optimal lower bound for 2-query locally decodable linear codes. *Inform. Process. Lett.*, 97(6):244–250, 2006.

[58] C. Slot and P. van Emde Boas. On tape versus core; An application of space efficient perfect hash functions to the invariance of space. In *Proc. of the 16th ACM Symp. on the Theory of Computing*, pages 391–400, 1984.

[59] R. Sprugnoli. Perfect hashing functions: A single probe retrieving method for static sets. *Commun. ACM*, 20(11):841–850, 1977.

[60] J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 357–371. Springer-Verlag, 1998.

[61] D. Stinson, T. van Trung, and R. Wei. Secure frameproof codes, key distribution patterns, group testing algorithms and related structures. *Journal of Statistical Planning and Inference*, 86(2):595–617, 2000.

[62] D. R. Stinson, R. Wei, and L. Zhu. New constructions for perfect hash families and related structures using combinatorial designs and codes. *J. of Combinatorial Designs*, 8:189–200, 2000.

[63] R. E. Tarjan and A. C. Yao. Storing a sparse table. *Commun. ACM*, 22(11):606–611, 1979.

[64] S. Wehner and R. de Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Proc. of the 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 1424–1436, 2005.

[65] D. Woodruff and S. Yekhanin. A geometric approach to information-theoretic private information retrieval. In *Proc. of the 20th IEEE Conf. on Computational Complexity*, pages 275–284, 2005.

[66] A. Yamamura and T. Saito. Private information retrieval based on the subgroup membership problem. In V. Varadharajan and Y. Mu, editors, *ACISP 2001*, volume 2119 of *Lecture Notes in Computer Science*, pages 206–220. Springer-Verlag, 2001.

[67] E. Y. Yang, J. Xu, and K. H. Bennett. Private information retrieval in the presence of malicious failures. In *Proc. of the 26th IEEE International Computer Software and Applications Conference*, pages 805–812, 2002.

A A Construction of a Minimal Perfect Hash Family

In this section we describe an (ℓ, k) minimal perfect hash family of size $\log \ell 2^{O(k)}$, namely we prove Claim 3.5. This construction is a combination of the constructions of Mehlhorn [48] and Slot and van Emde Boas [58], and uses techniques from [31]. The construction of the (ℓ, k) minimal hash family has four stages: first we construct an $(\ell, k, O(k^2 \log \ell))$ perfect hash family $\mathcal{H}_1$, second we construct an $(O(k^2 \log \ell), k, k^2)$ perfect hash family $\mathcal{H}_2$, then we construct a $(k^2, k, 6k)$ perfect hash family $\mathcal{H}_3$, and, finally, we construct a $(6k, k)$ minimal perfect hash family $\mathcal{H}_4$. The (ℓ, k) minimal hash family is

$$\mathcal{H} \stackrel{\text{def}}{=} \{h_4 \circ h_3 \circ h_2 \circ h_1 : h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, h_3 \in \mathcal{H}_3, h_4 \in \mathcal{H}_4\}.$$

That is, to reduce the range to k, we first reduce the range to $O(k^2 \log \ell)$, we, then, reduce it to k^2, then to $6k$, and finally to k.

We start by describing some results of [31] and [48]. We say that a function h shutters a set A if h restricted to A is one-to-one. The basic building block of the perfect hash family is the following construction:

Definition A.1 (The Function $h_a^{p,m}$) *For integers m, a and a prime p, define the function $h_a^{p,m} : [p-1] \to [m]$ where*

$$h_a(x) \stackrel{\text{def}}{=} ((ax \bmod p) \bmod m) + 1.$$

Furthermore, define the family of functions

$$\mathcal{H}_{p,m} \stackrel{\text{def}}{=} \{h_a^{p,m} : a \in [p-1]\}.$$

Claim A.2 ([31]) *Let k be an integer, p be a prime, and $A \subset [p-1]$ be a set of size k. The function $h_a^{p,2k^2}$ shutters A for at least half of all values a in $[p-1]$.*

Claim A.3 ([31]) *Let p be a prime, $A \subset [p-1]$, and $y \in [p-1]$. Define $B(A, a, y) \stackrel{\text{def}}{=} \left\{x \in A : h_a^{p,k} = y\right\}$. For every $A \subset [p-1]$, where $|A| = k$, there exists an $a \in [p-1]$ such that $\sum_{y \in [k]} |B(A, a, y)|^2 \leq 3k$.*

In the above claim, $B(A, a, y)$ is the set of $x \in A$ that are mapped to $y - 1$ by the function $(ax \bmod p) \bmod k$.

Claim A.4 ([48]) *For every integer ℓ and prime p define the function $\mathrm{MOD}_p^\ell : [\ell] \to [p]$, where*

$$\mathrm{MOD}_p^\ell(x) \stackrel{\text{def}}{=} (x \bmod p) + 1.$$

There exists a constant c that for every integers ℓ and k and every set $A \subset [\ell]$, where $|A| = k$, there is a prime $p < ck^2 \log \ell$ such that MOD_p^ℓ shutters A.

First stage. Let c be a the constant from Claim A.4. Define the family of functions

$$\mathcal{H}_1 \stackrel{\text{def}}{=} \left\{\mathrm{MOD}_p^\ell : p < ck^2 \log \ell \text{ is a prime } \right\}.$$

By Claim A.4, the family $\mathcal{H}_1$ is an $(\ell, k, ck^2 \ln \ell)$ perfect hash family of size $O(k^2 \log \ell)$.

Second stage. Let p_2 be a prime, where $ck^2 \log \ell < p_2 < 2ck^2 \log \ell$. Define the family of functions $\mathcal{H}_2 \stackrel{\text{def}}{=} \mathcal{H}_{p_2,2k^2}$. By Claim A.2, the family $\mathcal{H}_2$ is a $(ck^2 \log \ell, k, k^2)$ perfect hash family of size $O(k^2 \ln \ell)$.

Third Stage. This is the more complicated stage. First, we use the family $\mathcal{H}_{2k^2,k}$. This family *is not* a perfect hash family, however by Claim A.3, it hashes every set A of size k to different buckets, such that each bucket contains at most $O(\sqrt{k})$ elements. Let p_3 be a prime, where $2k^2 < p_3 \leq 4k^2$. For $a, a_1, \ldots, a_k \in [p_3 - 1]$ and a sequence $c_1, \ldots, c_k \in [6k]$, where $\sum_{y=1}^{k} c_y \leq 6k$, define the hash function $h_{a,a_1,\ldots,a_k,c_1,\ldots,c_k}$:

- $i \leftarrow ((ax \bmod p_3) \bmod k) + 1$.
- $j \leftarrow ((a_i x \bmod p_3) \bmod c_i) + 1$.
- Output $h_{a,a_1,\ldots,a_k,c_1,\ldots,c_k}(x) = \sum_{\ell=1}^{i-1} c_\ell + j$.

Now, define

$$\mathcal{H}_3 \stackrel{\text{def}}{=} \left\{ h_{a,a_1,\ldots,a_k,c_1,\ldots,c_k} : a, a_1, \ldots, a_k \in [p_3 - 1], \sum_{y=1}^{k} c_y \leq 6k \right\}.$$

We claim that $\mathcal{H}_3$ is a $(2k^2, k, 6k)$ perfect hash family of size $k^{O(k)}$. That is, for any set $A \subset [k^2]$ of size k, the following function $h_{a,a_1,\ldots,a_k,c_1,\ldots,c_k}$ shutters A: Let $a \in [p_3 - 1]$ be an element such that $\sum_{y \in [k]} |B(A, a, y)|^2 \leq 3k$; such a is guaranteed by Claim A.3. For $y \in [k]$, define $c_y \stackrel{\text{def}}{=} 2|B(A, a, y)|^2$, and let $a_y \in [p_3 - 1]$ be an element such that $h_{a_y}^{p_3,c_y}$ shutters $B(A, a, y)$; such a_y is guaranteed by Claim A.2.

We now explain the trick of [58] to reduce the size of the family $\mathcal{H}_3$ to $2^{O(k)}$ (compared to $k^{O(k)}$ above). As above, for any set $A \subset [k^2]$ of size k, let $a \in [p_3 - 1]$ be an element such that $\sum_{y \in [k]} |B(A, a, y)|^2 \leq 3k$. For $y \in [k]$, define $c_y \stackrel{\text{def}}{=} |B(A, a, y)|^2$. By Claim A.2, there is a single $b_1 \in [p_3 - 1]$ such that $h_{b_1}^{p_3,c_y}$ shutters the set $B(A, a, y)$ for at least half of all values y in $[k]$. Furthermore, there is a second $b_2 \in [p_3 - 1]$ such that $h_{b_2}^{p_3,c_y}$ shutters the set $B(A, a, y)$ for at least half of all values y in $[k]$ that are not shattered by b_1. After at most $\log k$ choices of a we shatter $B(A, a, y)$ for every $y \in [k]$. By proper encoding, this reduces the size of $\mathcal{H}_3$ to $2^{O(k)}$.

Forth stage. The final stage is a simple re-indexing. For every k distinct elements $x_1, x_2, \ldots, x_k$ in $[6k]$, define the function $h_{x_1,x_2,\ldots,x_k} : [6k] \to [k]$, where $h_{x_1,x_2,\ldots,x_k}(x_i) = i$ for $i \in [k]$ and $h_{x_1,x_2,\ldots,x_k}(x) = 1$ otherwise. Define the following family of functions

$$\mathcal{H}_4 \stackrel{\text{def}}{=} \{h_{x_1,x_2,\ldots,x_k} : 1 \leq x_1 < x_2 < \cdots x_k \leq 6k\}.$$

The family $\mathcal{H}_4$ is, clearly, a $(6k, k)$ minimal hash family of size $\binom{6k}{k} \leq 2^{6k}$.