Robust Information-Theoretic Private Information Retrieval

Amos Beimel[*] Yoav Stahl[†]

Abstract

A Private Information Retrieval (PIR) protocol allows a user to retrieve a data item of its choice from a database, such that the servers storing the database do not gain information on the identity of the item being retrieved. PIR protocols were studied in depth since the subject was introduced in Chor, Goldreich, Kushilevitz, and Sudan 1995. The standard definition of PIR protocols raises a simple question – what happens if some of the servers crash 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, i.e., protocols which still work correctly even if only k out of ℓ servers are available during the protocols' operation (the user does not know in advance which servers are available).

We present various robust PIR protocols giving different tradeoffs between the different parameters. These protocols are incomparable, i.e., for different values of n and k we will get better results using different protocols. We first present a generic transformation from regular PIR protocols to robust PIR protocols, this transformation is important since any improvement in the communication complexity of regular PIR protocol will immediately implicate improvement in the robust PIR protocol communication. We also present two specific robust PIR protocols. Finally, we present 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 databases.

1 Introduction

A Private Information Retrieval (PIR) protocol allows a user to retrieve a data item of his 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 value 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 [13], 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. This default setting can be easily extended to handle more general scenarios, e.g., of larger data items, several users, or several retrieved items per user.

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 (e.g., they have crashed or they are disconnected), 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 of whose complexity is $\binom{\ell}{k}$ times the complexity of the best known PIR protocol. Even

[*]Computer Science Dept., Ben-Gurion University, Beer-Sheva 84105, Israel. E-mail: beimel@cs.bgu.ac.il. Homepage: www.cs.bgu.ac.il/∼beimel.

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

for fairly small ℓ and k, the factor $\binom{\ell}{k}$ can 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 we need 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 motivation 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 hold the latest version of the database, then 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 can 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 that it receives, i.e., it need not wait for slow servers.

Related Work. Before proceeding, we give a brief overview of some relevant 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 the server is not allowed to gain *any* information about the retrieved bit, then the linear communication complexity of this solution is optimal [13]. To overcome this problem, [13] 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 $O(n^{1/3})$ bits [13], (2) a k-server protocol, for any constant $k > 1$, with communication complexity of $O(k^3 n^{1/(2k-1)})$ bits [17] (improving on [13, 3, 18], see also [7]), (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 [8], and (4) a protocol with $O(\log n)$ servers and communication complexity of $O(\log^2 n \log \log n)$ bits [5, 6, 13] . In all these protocols it is assumed that the servers do not communicate with each other. t-private protocols, in which the user is protected against collisions of up to t servers, have been considered in [13, 17, 7]. Specifically, the best communication complexity of such protocol is $O(n^{1/\lfloor (2k-1)/t \rfloor})$ [7]. For a more extensive discussion on PIR related work the reader can consult, e.g., [27].

One of the main tools we use is *perfect hash families* which were introduced by [21]. These families were first used in compiler design to prove lower bounds on the size of a computer program. In the last few years, perfect hash families have been applied to circuit complexity problems [23], derandomization of probabilistic algorithms [2], threshold cryptography [9, 11], and other tasks in cryptography [15, 28]. Perfect hash families are also considered from a combinatorial point of view [1, 4, 10, 12, 16, 19, 29]. A comprehensive overview on perfect hashing can be found in [14].

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 get better results using different protocols.

Our first result is a generic transformation from k-out-of-k PIR protocols to k-out-of-ℓ PIR protocols: we show that if there exists a perfect hash family $H_{\ell,k}$ of size $w_{\ell,k}$ (the definition of a perfect hash family appears in 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 is a generic transformation, any improvement in the communication complexity of k-out-of-k PIR protocols (e.g., the recent result of [8]) directly translates to improved robust PIR protocols. We also present a generic transformation from t-private k-out-of-k PIR protocols to 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, 13]. This protocol is a k-out-of-ℓ PIR protocol with communication complexity of $O(kn^{1/k}\ell \log \ell)$. That is, the communication in this protocol is polynomial in ℓ and k, however its dependency on n is worse than the protocols that can be obtained via the generic transformation.

Our third protocol combines Shamir's secret sharing scheme with the 2-server protocol of [13]. This results in a 2-out-of-ℓ protocol with communication complexity of $O(n^{1/3}\log \ell)$, 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. We next show that there exists 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. The communication complexity in the protocol is $O(kn^{1/\lfloor k/3\rfloor}\ell\log\ell)$.

Organization. In Section 2 we provide the necessary definitions, in Section 3 we show generic transformations from PIR protocols to robust PIR protocols, in Section 4 we show a specific construction of a k-out-of-ℓ robust PIR protocol, and in Section 5 we construct a 2-out-of-ℓ robust PIR protocol using Shamir's secret sharing scheme. Finally, in Section 6 we present robust PIR protocols tolerating Byzantine servers.

2 Preliminaries

We start with some notation. By $[k]$ we denote the set $\{1,\ldots,k\}$. Let GF(q) denote the finite field with q elements, where q is a prime-power. Given a vector $\vec{V}$, we denote $V[j]$ as the j-th coordinate of $\vec{V}$.

2.1 PIR Protocols

We define 1-round information-theoretic PIR protocols. 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 k-out-of-ℓ PIR protocol $\mathcal{P}=(\mathcal{R},\mathcal{Q},\mathcal{A},\mathcal{C})$ consists of a probability distribution $\mathcal{R}$ 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 any 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 $K=\{j_1,\ldots,j_k\}$. A k-out-of-ℓ protocol as above is a t-private robust PIR protocol, if it satisfies the following requirements:*

Correctness. *The user always computes the correct value of x_i from any k answers. Formally, for every $i\in\{1,...,n\}$, every random string r, every set $K=\{j_1,\ldots,j_k\}\subseteq\{1,\ldots,\ell\}$ and every database $x\in\{0,1\}^n$ it holds that, $\mathcal{C}(i,r,K,\mathcal{A}(j_1,\mathcal{Q}(j_1,i,r),x),\ldots,\mathcal{A}(j_k,\mathcal{Q}(j_k,i,r),x))=x_i$.*

t-**Privacy.** *Each collusion of up to t servers has no information about the bit that the user tries to retrieve: For every two indices $i_1,i_2\in\{1,\ldots,n\}$ and for every $\{j_1,\ldots,j_t\}\subseteq\{1,\ldots,\ell\}$, the distributions $\langle\mathcal{Q}(j_1,i_1,r),\ldots,\mathcal{Q}(j_t,i_1,r):r\in R\rangle$ and $\langle\mathcal{Q}(j_1,i_2,r),\ldots,\mathcal{Q}(j_t,i_2,r):r\in R\rangle$ are identical.*

We denote 1-private k-out-of-ℓ PIR protocol as k-out-of-ℓ PIR protocols. The main difference between the definition of PIR and robust PIR is in the correctness requirements. That is, the regular PIR protocols are k-out-of-k robust 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 $x\in\{0,1\}^n$, $i\in[n]$, and all random inputs. The* total communication *in the protocol is the* *number of bits communicated between the user and the ℓ servers.* The query complexity *is the maximal number of bits sent from the user to any single server, and the* answer complexity *is the maximal number of answer bits sent by any server.*

2.2 Shamir's Secret-Sharing

Secret-sharing schemes are an important tool in the construction of several PIR protocols. See [7] for a discussion on the role of secret sharing in PIR protocols. In general terms, a 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 [25] which we use in our protocols.

Shamir's scheme [25]. Let F be a finite field with $q > \ell$ elements, and let $\omega_1, \ldots, \omega_\ell$ be distinct nonzero elements of F. In order to share a secret $s \in F$ using a t-out-of-ℓ sharing scheme, the dealer chooses $t-1$ random elements $a_1, \ldots, a_{t-1}$, which together with the secret s define a univariate polynomial $p(Y) \overset{\text{def}}{=} a_{t-1}Y^{t-1} + a_{t-2}Y^{t-2} + \ldots + a_1 Y + s$. 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 $\alpha_{j_h} = \prod_{d \neq h} \frac{\omega_{j_d}}{\omega_{j_d} - \omega_{j_h}}$ 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 $\vec{V}$ of length m, i.e., $\vec{V} \in F^m$ we define the shares of the vector, denoted by $\langle \vec{V}_1, \ldots, \vec{V}_\ell \rangle$, where each $\vec{V}_j$ is a vector in F^m as follows: For each element $V[a]$, where $1 \leq a \leq m$, the user executes Shamir's t-out-of-ℓ secret sharing scheme independently over the field F producing ℓ shares $s_1^a, \ldots, s_\ell^a$. We then define the vector $\vec{V}_j$ as $\langle s_j^1, \ldots, s_j^m \rangle$, 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.

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

We start with 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.*

Proof: Let $\mathcal{P}$ be a 2-out-of-2 PIR protocol. Given the 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 comprises 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 received by each server guarantees 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 the queries $Q_m[0]$ and $Q_m[1]$ and thus he 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. 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. Since we assume that at least 2 servers are reachable, the user will receive answers from at least 2 servers, say server $\mathcal{S}_{j_1}$ and server $\mathcal{S}_{j_2}$ ($j_1 \neq j_2$). The binary representations of j_1 and j_2 differ in at least one bit; let m be the index of the first bit that differs between j_1 and j_2, and without loss of generality, $b_m^{j_1} = 0$ and $b_m^{j_2} = 1$. The user takes the answer received from server $\mathcal{S}_{j_1}$ for the query $Q_m[0]$ and the answer received from server $\mathcal{S}_{j_2}$ for the query $Q_m[1]$ and reconstruct the desired bit x_i.

This scheme is secure since each server receives only one query out of each pair of queries and these pairs of queries are independent. In this protocol each server receives $\log \ell$ queries and answers each one of them, thus the complexity of the protocol is the number of queries multiplied by $\mathrm{PIR}_2(n)$, i.e., the total communication is $O(\mathrm{PIR}_2(n)\ell \log \ell)$. □

Plugging the PIR protocol of [13] 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 show 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 [21]) *A* perfect hash family $H_{\ell,k} = \left\{h_1, \ldots, h_{w_{\ell,k}}\right\}$ *is a family of functions of the form: $h_a : \{1, \ldots, \ell\} \to \{1, \ldots, k\}$ such that for each subset $A \subseteq \{1, \ldots, \ell\}, |A| = k$, there exists an index a such that $|h_a(A)| = k$ (that is, h_a restricted to A is one-to-one and on-to). The size of the family is the number of functions in the family, denoted by $w_{\ell,k}$.*

We have 3 parameters for a perfect hash family – k, ℓ, and $w_{k,\ell}$. The parameters k and ℓ are part of the specification of the problem. On the other hand, we would like $w_{\ell,k}$ – the number of functions in the perfect hash family – to be as small as possible, since $w_{\ell,k}$ will directly affect our protocol's complexity.

Theorem 3.4 *If there exists a perfect hash family $H_{\ell,k}$ of size $w_{\ell,k}$ and if there exists 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, thus 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 comprises 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 the user holds the $w_{\ell,k}$ 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 completely 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 received by each server guarantees that if the user receives answers from at least k servers then it can reconstruct x_i.

Given a perfect hash family $H_{\ell,k}$, for each server $\mathcal{S}_j$ the user sends the following $w_{\ell,k}$ queries – for each $1 \leq a \leq w_{\ell,k}$, let $\Delta = h_a(j)$, then 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. In other words, the perfect hash family determines which queries we need to take from each vector of queries Q_a. 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 a permutation. We consider the answers $\{Q_a[h_a(j_1)], \ldots, Q_a[h_a(j_k)]\}$ received from these servers to the following queries. 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. □

The communication complexity of the above protocol depends on the size of the perfect hash family. The explicit hash family of [26] has size $\log \ell \cdot 2^{O(k)}$ (this is basically optimal [21]). Using the protocol of [13] and the hash family of [26] we get:

Corollary 3.5 *There is a k-out-of-ℓ protocol with total communication $2^{O(k)} n^{\frac{1}{2k-1}} \ell \log \ell$.*

Applying the protocol of [8] and the hash family of [26] we get:

Corollary 3.6 *There is a k-out-of-ℓ 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 "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.

Theorem 3.7 *If there is a perfect hash family $H_{\ell,k}$ of size $w_{\ell,k}$ and if there is 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 [7] and the hash family of [26] we get:

Corollary 3.8 *There is a t-private k-out-of-ℓ protocol with total communication $O(\ell \cdot \log \ell \cdot 2^{O(k)} \cdot n^{1/\lfloor (2k-1)/t \rfloor})$.*

3.3 A Generalized Transformation

We now show a generalization of the previous transformation, where our goal is to reduce the dependency on k. As seen in [21] the size of every perfect hash family $H_{\ell,k}$ is at least 2^k, thus, we first generalize the notion of perfect hashing.

Definition 3.9 *An α-perfect hash family $H_{\ell,k,\alpha} = \left\{ h_1, \ldots, h_{w_{\ell,k,\alpha}} \right\}$ (where $\alpha \leq 1$) is a family of functions $h_a : \{1, \ldots, \ell\} \to \{1, \ldots, \lfloor \alpha k \rfloor\}$ such that for each subset $A \subseteq \{1, \ldots, \ell\}, |A| = k$, there exists an index a such that $|h_a(A)| = \lfloor \alpha k \rfloor$.*

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

Theorem 3.10 *If there is an α-perfect hash family $H_{k,\ell,\alpha}$ of size $w_{\ell,k,\alpha}$ and if there is an $\lfloor \alpha k \rfloor$-out-of-$\lfloor \alpha k \rfloor$ PIR protocol 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 the one shown in 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. Let $\mathcal{S}_{j_1}, \ldots, \mathcal{S}_{j_k}$ be k servers from which the user receives answers. Using the α-perfect hash 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 $\lfloor \alpha k \rfloor$ distinct 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. $\square$

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 decrease as k gets bigger and since we are using $\alpha < 1$ then 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 get a family whose size is small.

Claim 3.11 *There exists an $\frac{1}{\ln k}$-perfect hash family of size $O(\frac{k \log \ell}{\log \log k})$.*

Proof: We will prove the claim using a probabilistic proof. As a first step lets consider a specific subset $A \subseteq \{1, \ldots, \ell\}$, $|A| = k$, 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 $c \in \{1, \ldots, \lfloor \alpha k \rfloor\}$. We now look at the probability

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

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

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

Therefore,

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

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 α-perfect hash family is smaller then 1, i.e., there exists an α-perfect hash family of size $w_{\ell,k,\alpha}$. Thus, it suffices that $\ell^k < (\ln k)^{w_{\ell,k,\alpha}}$, i.e., $w_{\ell,k,\alpha} > \frac{k \log \ell}{\log \log k}$. $\square$

In the above analysis, Inequality (1) could have been derived from the so-called coupon collector problem, see, e.g., [22, 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 describe a more efficient transformation to robust PIR protocol using the $\frac{1}{\ln k}$-perfect hash family. If, for example, there exists a PIR protocol with communication $\mathrm{poly}(k) \cdot n^{O(\frac{1}{k \log k})}$ then we will get a robust protocol with communication complexity of $\mathrm{poly}(k, \ell) \cdot n^{\frac{1}{2k}}$. Notice that the recent PIR protocols [8] 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 which uses the polynomial interpolation based PIR Protocol of [5, 6, 13]. We start with a technical lemma, and then present the protocol.

Lemma 4.1 *Let d and m be integers such that $m = \Omega(d \cdot n^{\frac{1}{d}})$. There is a function $E : \{1, \ldots, n\} \longrightarrow \{0,1\}^m$ and an m-variant degree d polynomial P_x such that in every field $P_x(E(i)) = x_i$ 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, let $E(i)_a$ be the a-th bit of $E(i)$ and define $P_x(Z_1, \ldots, Z_m) \overset{\text{def}}{=} \sum_{i=1}^{n} x_i \prod_{E(i)_a = 1} Z_a$. $\square$

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 the retrieval index i, the user does the following: Calculates the vector $E(i) = \langle y_1, \dots y_m \rangle$, where y_j is the j-th bit of $E(i)$. Now the user uses Shamir's 2-out-of-ℓ scheme [25], over a finite field with $O(\ell)$ elements, to share $E(i)$. That is, it chooses at random m polynomials $\{p_1, \dots, p_m\}$ (each of degree 1) such that $p_a(0) = y_a$ for each $1 \le a \le m$. Let $\omega_1, \dots, \omega_\ell$ be distinct nonzero elements of the field, the user sends to server $\mathcal{S}_j$ the shares $\langle p_1(\omega_j), \dots, p_m(\omega_j) \rangle$.

We now consider the following univariate polynomial: $R(Y) = P_x(p_1(Y), \dots, p_m(Y))$; the degree of this polynomial 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, $R(0) = P_x(p_1(0), \dots, p_m(0)) = P_x(y_1, \dots, y_m) = x_i$. Furthermore, the server $\mathcal{S}_j$ can compute $R(\omega_j)$ without knowing R since $R(\omega_j) = P_x(p_1(\omega_j), \dots, p_m(\omega_j))$ and since $\langle p_1(\omega_j), \dots, p_m(\omega_j) \rangle$ 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.

The user upon receiving any k answers $R(\omega_{j_1}), \dots, R(\omega_{j_k})$ (from k different servers) reconstructs the polynomial R by interpolation (since the user has k points on a polynomial of degree $d = k-1$) 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 the secret $E(i)$ in a 2-out-of-ℓ secret sharing scheme. Thus, the protocol is private. 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 $O(m\ell \log \ell) = O(kn^{\frac{1}{k-1}} \ell \log \ell)$. □

In the above protocol, the answer complexity is larger than the query complexity. We will balance these complexities using the balancing technique of [13], 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_1, i_2 is the 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 that it got for the row i_1 and reconstructs x_{i_1,i_2}. The user sends one query to each server with database of size $\alpha(n)$, 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 is 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: $\log \ell \cdot \frac{n}{\alpha(n)} = \log \ell k \alpha(n)^{\frac{1}{k-1}}$. Taking $\alpha(n) = O(n^{\frac{k-1}{k}})$, yields 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 $O(\frac{k}{t} n^{\frac{1}{\lfloor (k-1)/t \rfloor}} \log \ell)$ and answer complexity of $O(\log \ell)$ per server.*

Proof: Let $d = \left\lfloor \frac{k-1}{t} \right\rfloor$, $m = \Theta(dn^{\frac{1}{d}})$, 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 $p_1, \dots, p_m$ is t. Thus, the degree of R is $dt = \lfloor \frac{k-1}{t} \rfloor \cdot t \le k-1$. The user, upon receiving k answers $R(\omega_{j_1}), \dots, R(\omega_{j_k})$ (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 $R(0) = x_i$.

Next we analyze the properties of the protocol. A coalition of t servers does not gain any information on i, since the user uses Shamir's $(t+1)$-out-of-ℓ secret sharing scheme. Thus, the protocol is t-private. As for the communication, the user sends m shares to each server, each share is of size $O(\log \ell)$. Each server sends an answer of length $O(\log \ell)$ thus the query complexity is $O(m \log \ell) = O(\frac{k}{t} n^{\frac{1}{\lfloor (k-1)/t \rfloor}} \log \ell)$ per server. $\square$

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 same technique described in 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: $\log \ell \cdot \frac{n}{\alpha(n)} = \frac{k}{t} \log \ell \cdot \alpha(n)^{\frac{1}{\lfloor (k-1)/t \rfloor}}$. 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. $\square$

5 A Robust PIR Protocols Using Shamir's Secret Sharing

In this section we show how one can use Shamir's secret sharing in order 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 computes the vectors $\langle \vec{V}_1, \ldots, \vec{V}_\ell \rangle$ – the shares of Shamir's scheme (as described in Section 2.2) over GF($2^{\lceil \log \ell \rceil}$) of the unit vector $\vec{e}_i$ of length n, and sends $\vec{V}_j$ to server $\mathcal{S}_j$ for each $1 \le j \le \ell$. Server $\mathcal{S}_j$, upon receiving $\vec{V}_j$, sends back to the user the following scalar multiplication: $a_j \overset{\text{def}}{=} \vec{V}_j \cdot \vec{x}$, i.e., the server computes the scalar product of the database and the vector $\vec{V}_j$ and sends the result to the user.

The user upon receiving k answers $a_{j_1}, \ldots, a_{j_k}$ uses the appropriate constants $\alpha_{j_1}, \ldots, \alpha_{j_k}$ (see Section 2.2) to perform the following computation (in the following proof all additions and multiplications are done in the GF($2^{\lceil \log \ell \rceil}$)):

$$\sum_{h=1}^{k} \alpha_{j_h} a_{j_h} = \sum_{h=1}^{k} \alpha_{j_h} (\vec{V}_{j_h} \cdot \vec{x}) = (\sum_{h=1}^{k} \alpha_{j_h} \vec{V}_{j_h}) \cdot \vec{x} = \vec{e}_i \cdot \vec{x} = x_i.$$

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

We now present a more efficient protocol that uses the above ideas combined with the 2-server protocol of [13]. This 2-out-of-ℓ protocol works with total communication of $O(n^{1/3} \ell \log \ell)$. Lets first recall the protocols presented by [13]:

ORIGINAL PROTOCOL (VARIANT OF [13]). 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 $i_r \in [n^{1/3}]$ for $r = 1, 2, 3$. This is done using the natural mapping from $\{0,1\}^n$ to $(\{0,1\}^{n^{1/3}})^3$. 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, thus cancels itself. Therefore, the user outputs x_{i_1,i_2,i_3} as required. Furthermore, the communication is $O(n^{1/3})$.

We can look at $\vec{A}_r^2 = \vec{A}_r^1 \oplus \vec{e}_{i_r}$ and $\vec{A}_r^1$ as two shares in a 2-out-of-2 sharing scheme of the unit vector $\vec{e}_{i_r}$. 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 $\vec{e}_{i_r}$; these shares are used to generate the queries for the protocol.

We next define some notation concerning cubes. This notation is helpful in describing the next protocols.

The Two Server Protocol of [13]

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

   The user sends $\vec{A}_1^j, \vec{A}_2^j, \vec{A}_3^j$ to $\mathcal{S}_j$ for $j = 1,2$.

2. Server $\mathcal{S}_j$ computes for every $b \in [n^{1/3}]$

$$a_{1,b}^j \stackrel{\text{def}}{=} \vec{A}_2^j \cdot x_{b,*,*} \cdot \vec{A}_3^j, \qquad a_{2,b}^j \stackrel{\text{def}}{=} \vec{A}_1^j \cdot x_{*,b,*} \cdot \vec{A}_3^j \text{ and}$$
$$a_{3,b}^j \stackrel{\text{def}}{=} \vec{A}_1^j \cdot x_{*,*,b} \cdot \vec{A}_2^j,$$

   and sends the $3n^{1/3}$ bits $\left\{a_{r,b}^j \ : \ r \in \{1,2,3\}\,,\ b \in [n^{1/3}]\right\}$ to the user.

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

Figure 1: The two server protocol of [13] with communication $O(n^{1/3})$.

Definition 5.1 *Let $x = (\{0,1\}^m)^3$ be a three dimensional cube. We denote $x_{j_1,*,*}$ as the matrix of all the elements of x where the first index of this element 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 $x_{j_1,j_2,*}$ as the vector obtained from the 3-dimensional cube x by taking all the elements of x where the first index of this element is j_1 and the second is j_2. Formally, we define $x_{j_1,j_2,*}$ as the vector $\vec{A}$ where $\vec{A}_{i_1} = x_{j_1,j_2,i_1}$. We define x_{*,j_1,j_2} and $x_{j_1,*,j_2}$ similarly.*

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:

Given ℓ and the retrieval index $i = \langle i_1, i_2, i_3 \rangle$, the user computes the vector $\langle \vec{U}_1, \ldots, \vec{U}_\ell \rangle$, the vector $\langle \vec{V}_1, \ldots, \vec{V}_\ell \rangle$, and the vector $\langle \vec{W}_1, \ldots, \vec{W}_\ell \rangle$ as the shares in a Shamir's 2-out-of-ℓ scheme of the unit vector $\vec{e}_{i_1}$, the unit vector $\vec{e}_{i_2}$, and the unit vector $\vec{e}_{i_3}$ respectively. The user sends the query $\vec{U}_j, \vec{V}_j, \vec{W}_j$ to server $\mathcal{S}_j$ for each $1 \le j \le \ell$.

Server $\mathcal{S}_j$ upon receiving $\vec{U}_j, \vec{V}_j, \vec{W}_j$ sends back to the user the following $3n^{1/3}$ numbers: For each $1 \le a \le n^{1/3}$ the server sends to the user: The set of numbers – $\vec{V}_j \cdot x_{a,*,*} \cdot \vec{W}_j$, the set of numbers – $\vec{U}_j \cdot x_{*,a,*} \cdot \vec{W}_j$, and the set of numbers $\vec{U}_j \cdot x_{*,*,a} \cdot \vec{V}_j$, 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 r and q considers the following 6 numbers:

$$\vec{V}_r \cdot x_{i_1,*,*} \cdot \vec{W}_r,\ \vec{U}_r \cdot x_{*,i_2,*} \cdot \vec{W}_r,\ \vec{U}_r \cdot x_{*,*,i_3} \cdot \vec{V}_r, \vec{V}_q \cdot x_{i_1,*,*} \cdot \vec{W}_q,\ \vec{U}_q \cdot x_{*,i_2,*} \cdot \vec{W}_q,\ \vec{U}_q \cdot x_{*,*,i_3} \cdot \vec{V}_q.$$

The following claim is similar to the fact that in the protocol of [13] the user reconstructs the correct bit x_i.

Claim 5.3 *There exists a linear combination of these 6 numbers that computes to the desired bit x_{i_1,i_2,i_3}.*

Proof: In our proof we use the constants from Shamir's scheme α_r and α_q (see Section 2.2), these two constants are independent of the answers received from the servers.[1] Denote $\dot{\vec{U}} \stackrel{\text{def}}{=} \alpha_r \vec{U}_r$ and $\ddot{\vec{U}} \stackrel{\text{def}}{=} \alpha_q \vec{U}_q$. We

[1]The constants α_r and α_q depend both on r and q, which means that the servers themselves cannot compute them since the servers do not know in advance which of the servers will send an answer to the user.

denote $\dot{\vec{W}}, \ddot{\vec{W}}$ and $\dot{\vec{V}}, \ddot{\vec{V}}$ similarly. Thus,

$$\ddot{\vec{U}} + \dot{\vec{U}} = \vec{e}_{i_1}, \; \ddot{\vec{V}} + \dot{\vec{V}} = \vec{e}_{i_2}, \text{ and } \ddot{\vec{W}} + \dot{\vec{W}} = \vec{e}_{i_3}. \tag{2}$$

Notice that $\dot{\vec{V}} \cdot x_{i_1,*,*} \cdot \dot{\vec{W}} = (\alpha_r)^2(\vec{V}_r \cdot x_{i_1,*,*} \cdot \vec{W}_r)$. In our computation we multiply each of the first three numbers by α_r^2 and each of the last three numbers by α_q^2 and consider the following combination:

$$\begin{aligned} S \;\overset{\text{def}}{=}\; & \dot{\vec{V}} \cdot x_{i_1,*,*} \cdot \dot{\vec{W}} + \dot{\vec{U}} \cdot x_{*,i_2,*} \cdot \dot{\vec{W}} + \dot{\vec{U}} \cdot x_{*,*,i_3} \cdot \dot{\vec{V}} \\ & + \ddot{\vec{V}} \cdot x_{i_1,*,*} \cdot \ddot{\vec{W}} + \ddot{\vec{U}} \cdot x_{*,i_2,*} \cdot \ddot{\vec{W}} + \ddot{\vec{U}} \cdot x_{*,*,i_3} \cdot \ddot{\vec{V}}. \end{aligned}$$

We use the fact that in GF$(2^{\lceil \log \ell \rceil})$ the sum of every number with itself is zero, so we add the number $\ddot{\vec{V}} \cdot x_{i_1,*,*} \cdot \dot{\vec{W}}$ twice, the number $\dot{\vec{U}} \cdot x_{*,i_2,*} \cdot \ddot{\vec{W}}$ twice, and the number $\ddot{\vec{U}} \cdot x_{*,*,i_3} \cdot \dot{\vec{V}}$ twice and get:

$$\begin{aligned} S \;=\; & \dot{\vec{V}} \cdot x_{i_1,*,*} \cdot \dot{\vec{W}} + \ddot{\vec{V}} \cdot x_{i_1,*,*} \cdot \dot{\vec{W}} + \dot{\vec{U}} \cdot x_{*,i_2,*} \cdot \dot{\vec{W}} + \dot{\vec{U}} \cdot x_{*,i_2,*} \cdot \ddot{\vec{W}} \\ & + \dot{\vec{U}} \cdot x_{*,*,i_3} \cdot \dot{\vec{V}} + \ddot{\vec{U}} \cdot x_{*,*,i_3} \cdot \dot{\vec{V}} + \ddot{\vec{V}} \cdot x_{i_1,*,*} \cdot \ddot{\vec{W}} + \ddot{\vec{V}} \cdot x_{i_1,*,*} \cdot \dot{\vec{W}} \\ & + \ddot{\vec{U}} \cdot x_{*,i_2,*} \cdot \ddot{\vec{W}} + \dot{\vec{U}} \cdot x_{*,i_2,*} \cdot \ddot{\vec{W}} + \ddot{\vec{U}} \cdot x_{*,*,i_3} \cdot \dot{\vec{V}} + \ddot{\vec{U}} \cdot x_{*,*,i_3} \cdot \ddot{\vec{V}} \\ =\; & (\dot{\vec{V}} + \ddot{\vec{V}}) \cdot x_{i_1,*,*} \cdot \dot{\vec{W}} + \dot{\vec{U}} \cdot x_{*,i_2,*} \cdot (\dot{\vec{W}} + \ddot{\vec{W}}) \\ & + (\dot{\vec{U}} + \ddot{\vec{U}}) \cdot x_{*,*,i_3} \cdot \dot{\vec{V}} + \; \ddot{\vec{V}} \cdot x_{i_1,*,*} \cdot (\ddot{\vec{W}} + \dot{\vec{W}}) \\ & + (\ddot{\vec{U}} + \dot{\vec{U}}) \cdot x_{*,i_2,*} \cdot \ddot{\vec{W}} + \ddot{\vec{U}} \cdot x_{*,*,i_3} \cdot (\ddot{\vec{V}} + \dot{\vec{V}}) \\ =\; & \vec{e}_{i_2} \cdot x_{i_1,*,*} \cdot \dot{\vec{W}} + \dot{\vec{U}} \cdot x_{*,i_2,*} \cdot \vec{e}_{i_3} \\ & + \vec{e}_{i_1} \cdot x_{*,*,i_3} \cdot \dot{\vec{V}} + \ddot{\vec{V}} \cdot x_{i_1,*,*} \cdot \vec{e}_{i_3} \\ & + \vec{e}_{i_1} \cdot x_{*,i_2,*} \cdot \ddot{\vec{W}} + \ddot{\vec{U}} \cdot x_{*,*,i_3} \cdot \vec{e}_{i_2}. \end{aligned}$$

The last equality follows (2). Notice that $\vec{e}_{i_2} \cdot x_{i_1,*,*} = \vec{e}_{i_1} \cdot x_{*,i_2,*} = x_{i_1,i_2,*}$ and similarly $x_{*,i_2,*} \cdot \vec{e}_{i_3} = x_{*,*,i_3} \cdot \vec{e}_{i_2} = x_{*,i_2,i_3}$ and $x_{i_1,*,*} \cdot \vec{e}_{i_3} = \vec{e}_{i_1} \cdot x_{*,*,i_3} = x_{i_1,*,i_3}$ (multiplication from the right replaces the rightmost $*$ and multiplication from the left replaces the leftmost $*$), thus we get:

$$\begin{aligned} S \;=\; & x_{i_1,i_2,*} \cdot (\dot{\vec{W}} + \ddot{\vec{W}}) + (\dot{\vec{U}} + \ddot{\vec{U}}) \cdot x_{*,i_2,i_3} + x_{i_1,*,i_3} \cdot (\dot{\vec{V}} + \ddot{\vec{V}}) \\ =\; & x_{i_1,i_2,*} \cdot \vec{e_{i_3}} + \vec{e_{i_1}} \cdot x_{*,i_2,i_3} + x_{i_1,*,i_3} \cdot \vec{e_{i_2}} = x_{i_1,i_2,i_3} + x_{i_1,i_2,i_3} + x_{i_1,i_2,i_3} \;=\; x_{i_1,i_2,i_3}. \end{aligned}$$

□

We now provide the proof of the protocol's privacy and its communication complexity analysis. 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$(2^{\lceil \log \ell \rceil})$ 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 can 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 might be malicious servers or have a corrupted or obsolete database, these servers can return any answer to the user's query. 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 $\mathcal{P} = (\mathcal{R}, \mathcal{Q}, \mathcal{A}, \mathcal{C})$ 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 $i \in \{1, ..., n\}$, every random string r, every set $K = \{j_1, \ldots, j_k\} \subseteq \{1, \ldots, \ell\}$, every database $x \in \{0,1\}^n$, and every k answers $\{a_1, \ldots, a_k\}$ such that $|\{a_w : a_w = \mathcal{A}(j_w, \mathcal{Q}(j_w, i, r), x)\}| \geq k - b$, we get that: $C(i, r, K, a_1, \ldots, a_k) = x_i$.*

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

The first observation is that if the server 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 $(k-a)/2$ Byzantine-robust k-out-of-ℓ PIR protocol with total communication $\mathrm{PIR}_a^\ell(n)$.*

Proof: The user and the servers execute an a-out-of-ℓ robust 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 maximal 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 x_i.

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

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 problem, this is a k-out-of-ℓ robust 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 6.3 *There is a $\frac{k}{3}$ 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 protocol described in Section 4. In this protocol, the answers of the honest servers are points on a univariate polynomial R whose degree $\lfloor k/3 \rfloor - 1$. 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 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. Next, we show that at most $2\lfloor k/3 \rfloor - 1$ points from the answers lie on a polynomial B of degree at most $\lfloor k/3 \rfloor - 1$ which is not R: Since B and R are different polynomials of degree at most $\lfloor k/3 \rfloor - 1$, they can have at most $\lfloor k/3 \rfloor - 1$ common points. Since the honest servers send points on R, they can contribute at most $\lfloor k/3 \rfloor - 1$ points on B. Furthermore, the Byzantine parties can contribute at most $\lfloor k/3 \rfloor$ points on B, thus we get at most $2\lfloor k/3 \rfloor - 1 < \lceil 2k/3 \rceil$ points that lie on B.

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. The user does this task using the decoding algorithm of the Reed-Solomon error correcting codes [24]. (For more information on error-correcting codes the reader can refer to, e.g., [20].) The communication complexity of the above protocol is the communication complexity of the $\lfloor k/3 \rfloor$-out-of-ℓ protocol described in Section 4, i.e., $O(kn^{1/\lfloor k/3 \rfloor}\ell \log \ell)$. $\square$

Since we consider Byzantine servers, the assumption that they do not cooperate is questionable, thus it might be more reasonable to consider b-private robust PIR protocols in the presence of b Byzantine servers. That is, 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 6.4 *Let a be a parameter where $k/3 < a \leq k$, and define $b = (k-a)/2$. Assume there is a b-private a-out-of-ℓ robust 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 b-private a-out-of-ℓ PIR protocol with sub-linear communication exists only if $a \geq b$, thus we get $a \geq k/3$.

Corollary 6.5 *There is a t-private t Byzantine-robust k-out-of-ℓ PIR protocol (where $t < k/3$) with total communication $O(\frac{k}{3t} n^{\frac{1}{\lfloor (k-1)/3t \rfloor}} \ell \log \ell)$.*

The idea is to use the t-private $\lfloor k/3 \rfloor$-out-of-ℓ protocol described in Lemma 4.4 in the same way as in Theorem 6.3.

Acknowledgements. We thank Eyal Kushilevitz for helpful discussions.

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 *Proc. of the 24th International Colloquium on Automata, Languages and Programming*, volume 1256 of *Lecture Notes in Computer Science*, pages 401–407, 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 Annu. 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] 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, 2001.

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

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

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

[11] 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.

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

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

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

[15] 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.

[16] M. L. Fredman and J. Komlos. On the size of separating systems and families of perfect hash funtions. *SIAM J. Alg. Discrete Methods*, 5:61–68, 1984.

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

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

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

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

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

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

[23] 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.

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

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

[26] 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 Annu. ACM Symp. on the Theory of Computing*, pages 391–400, 1984.

[27] Y. Stahl. Robust information-theoretic private information retrieval. Master's thesis, Ben-Gurion University, Beer-Sheva, 2002.

[28] 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.

[29] 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.