PIR Array Codes with Optimal PIR Rate

SIMON BLACKBURN*    TUVI ETZION†

July 7, 2016

Abstract

There has been much recent interest in Private information Retrieval (PIR) in models where a database is stored across several servers using coding techniques from distributed storage, rather than being simply replicated. In particular, a recent breakthrough result of Fazelli, Vardy and Yaakobi introduces the notion of a PIR array code, and uses this notion to produce efficient protocols.

In this paper we are interested in designing PIR array codes with good parameters. We consider the case when we have $m$ servers, with each server storing a fraction $(1/s)$ of the bits of the database; here $s$ is a fixed rational number with $s > 1$. We study the maximum PIR rate of a PIR array code with the $k$-PIR property, where the PIR rate is defined to be $k/m$. We present upper bounds on the achievable rate, some constructions, and ideas how to obtain PIR array codes with high PIR rate. In particular, we present constructions that asymptotically meet our upper bounds, and the exact largest PIR rate is obtained when $1 < s \leq 2$. Most, if not all, of our constructions will make use of set systems, i.e. some type of block design.

*Department of Mathematics, Royal Holloway College, Egham, Surrey TW20 0EX, United Kingdom e-mail: S.Blackburn@rhul.ac.uk.
†Department of Computer Science, Technion, Haifa 3200003, Israel, e-mail: etzion@cs.technion.ac.il. The research was performed while the second author visited Royal Holloway University of London under EPSRC Grant EP/N022114/1.
1 Introduction

A Private Information Retrieval (PIR) protocol allows a user to retrieve a data item from a database, in such a way that the servers storing the data will get no information about which data item was retrieved. The problem was introduced in [4]. The protocol to achieve this goal assumes that the servers are curious but honest, so they don’t collude. It is also assumed that the database is error-free and synchronized all the time. For a set of $k$ servers, the goal is to design a $k$-server PIR protocol, in which the efficiency of the PIR is measured by the total number of bits transmitted by all parties involved. This model is called an information-theoretic PIR; there is also computational PIR, in which the privacy is defined in terms of the inability of a server to compute which file was retrieved in reasonable time [22]. In this paper we will be concerned only with information-theoretic PIR.

The classical model of Private Information Retrieval assumes that each server stores a copy of an $n$-bit database, so the storage overhead, namely the total number of bits stored by all servers, is $nk$. However, recent work combines PIR protocols with techniques from distributed storage (where each server stores only some of the database) to reduce the storage overhead. This approach was first considered in [27], and several papers developed this direction further [1, 3, 11, 28, 29]. Our discussion will follow the breakthrough approach presented by Fazeli, Vardy, and Yaakobi [12, 13], which shows that $m$ servers (for some $m > k$) may emulate a $k$-server PIR protocol with storage overhead significantly lower than $nk$.

Fazeli et al introduce the key notion of a $[t \times m, p]$ k-PIR array code, which is defined as follows. Let $x_1, x_2, \ldots, x_p$ be a basis of a vector space of dimension $p$ (over some finite field $\mathbb{F}$). A $[t \times m, p]$ array code is simply a $t \times m$ array, each entry containing a linear combination of basis elements $x_i$. A $[t \times m, p]$ array code satisfies the k-PIR property (or is a $[t \times m, p]$ k-PIR array code) if for every $i \in \{1, 2, \ldots, p\}$ there exist $k$ pairwise disjoint subsets $S_1, S_2, \ldots, S_k$ of columns so that for all $j \in \{1, 2, \ldots, k\}$ the element $x_i$ is contained in the linear span of the entries of the columns $S_j$. The following example of a (binary) $[7 \times 4, 12]$ 3-PIR array code is taken from [13]:

| $x_1$ | $x_2$ | $x_3$ | $x_1 + x_2 + x_3$ |
|-------|-------|-------|--------------------|
| $x_2$ |       |       |                    |
| $x_4$ | $x_5$ | $x_4 + x_5 + x_6$ | $x_4$               |
| $x_5$ |       | $x_6$ | $x_8$              |
| $x_7$ | $x_7 + x_8 + x_9$ | $x_9$ | $x_7$              |
| $x_8$ |       | $x_{10}$ | $x_{11}$ | $x_{12}$ |
| $x_{10} + x_{11} + x_{12}$ | $x_{11}$ | $x_{12}$ | $x_{10}$ |

The 3-PIR property means that for all $j \in \{1, 2, \ldots, 12\}$ we can find 3 disjoint subsets of columns whose entries span a subspace containing $x_j$. For example, $x_5$ is in the span of the entries in the subsets $\{1\}$, $\{2\}$ and $\{3, 4\}$ of columns; $x_{11}$ is in the span of the entries in the subsets $\{1, 4\}$, $\{2\}$ and $\{3\}$ of columns.

In the example above, many of the entries in the array consist of a single basis element; we call such entries singletons.

Fazeli et al use a $[t \times m, p]$ $k$-PIR array code as follows. The database is partitioned into $p$ parts $x_1, x_2, \ldots, x_p$, each part encoded as an element of the finite field $\mathbb{F}$. Each of a set of $m$
servers stores $t$ linear combinations of these parts; the $j$th server stores linear combinations corresponding to the $j$th column of the array code. We say that the $j$th server has $t$ cells, and stores one linear combination in each cell. The authors show that the $k$-PIR property of the array code allows the servers to emulate all known efficient $k$-server PIR protocols. But the storage overhead is $ntm/p$, and this can be significantly smaller than $nk$ if a good array code (namely one where $tm/p$ is much smaller than $k$) is used. Define $s = p/t$, so $s$ is the reciprocal of the ratio between the length of the database and the length of the data stored on each server. For small storage overhead, we would like the ratio

$$\frac{nk}{ntm/p} = \frac{k}{m}$$

to be as large as possible. We define the PIR rate (rate in short) of a $[t \times m, p]$ $k$-PIR array code to be $k/m$. In applications, we would like the rate to be as large as possible for several reasons: when $s$, which represents the amount of storage required at each server, is fixed such schemes give small storage overhead; we wish to use a minimal number $m$ of servers, so $m$ should be as small as possible; large values of $k$ are desirable, as they lead to protocols with lower communication complexity. We define $g(s, t)$ to be the largest rate of a $[t \times m, p]$ $k$-PIR array code when $s$ and $t$ (and so $p$) are fixed. We define $g(s) = \lim_{t \to \infty} g(s, t)$.

Most of the analysis in [12, 13] was restricted to the case $t = 1$, in which case we say that we have (the generator matrix for) a $k$-PIR code of length $m$ and dimension $p$.

The following two results presented in [13] are the most relevant for our discussion. The first one corresponds to the case where each server holds a single cell, i.e. this is a PIR code (not an array code with $t > 1$).

**Theorem 1.** For any given positive integer $s$, $g(s, 1) \leq (2^s - 1)/(2^s - 1)$, with equality if and only if $k$ is divisible by $2^s - 1$.

The second result is a consequence of the only construction of PIR array codes given [13] which is not an immediate consequence of the constructions for PIR codes; this construction will be discussed below.

**Theorem 2.** For any integer $s \geq 3$, we have $g(s, s - 1) \geq s/(2s - 1)$.

The goal of this paper is first to improve on the result of Theorem 2. This will be done by providing several new constructions for $k$-PIR array codes which will imply lower bounds on $g(s, t)$ for a large range of pairs $(s, t)$. This will immediately imply a related bound on $g(s)$ for various values of $s$. Contrary to the construction in [13], the value of $s$ in our constructions is not necessarily an integer (this possible feature was mentioned in [13]: each rational number greater than one will be considered. We will also provide various upper bounds on $g(s, t)$, and related upper bounds on $g(s)$. It will be proved that some of the upper bounds on $g(s, t)$ are tight and also our main upper bound on $g(s)$ is tight.

We comment that, surprisingly, the $k$-PIR property of a $k$-PIR code is exactly what is needed in another concept of distributed storage, namely locality with availability [13]. Locally repairable codes (LRCs) are erasure codes which allow local correction of erasures, where any code symbol can be recovered by using a given small number of other code symbols. Such LRCs were introduced in [14]. Since data stored in distributed storage media
might need to be read by several users in modern applications, we might need to produce
the same item in several disjoint ways; this property is called \textit{availability}. Codes which
provide both locality and availability were first proposed in \cite{26}. More research work on this
new topic of research can be found also in \cite{19, 23, 30, 31, 34} The connection between PIR
codes and codes with availability immediately give new constructions and bounds for codes
with availability. We note that the two concepts are not exactly the same, since codes with
locality are required to be systematic, something which is not required from PIR codes.

To summarise our notation used in the remainder of the paper:

1. \( n \) - the number of bits in the database.

2. \( p \) - number of parts the database is divided into. The parts will be denoted by
\( x_1, x_2, \ldots, x_p \).

3. \( \frac{1}{s} \) - the fraction of the file which is stored in a server.

4. \( m \) - the number of servers (or the number of columns in the array).

5. \( t \) - number of cells in a server (or the number of rows in the array); so \( t = p/s \).

6. \( k \) - the array code allows the servers to emulate a \( k \)-PIR protocol.

7. \( g(s, t) \) - the largest PIR rate of a \([t \times m, p]\) \( k \)-PIR array code.

8. \( g(s) = \lim_{t \to \infty} g(s, t) \).

Though a PIR array code is formally an array of vectors, we use terminology carried over
from the application we have in mind. So we refer to a column of this array as a server, and
an entry of this column as a cell.

The information in the \( t \) cells of a given server spans a subspace \( V \) of \( \mathbb{F}^p \) whose dimension
is at most \( t \). It is this subspace, rather than the values in individual cells of the server, which
is important for the \( k \)-PIR property. Changing the cells to produce a new spanning set for
\( V \), or even to replace \( V \) by a larger subspace containing it, cannot harm the \( k \)-PIR property.
So, since the \( x_i \) are linearly independent, without loss of generality we can (and do) make
two assumptions in our analysis and constructions:

- if \( x_i \) can be derived from information in certain server alone, the singleton \( x_i \) is stored
  as the value of one of the cells of this server;

- the data stored in any server’s cells are linearly independent, i.e. the subspace spanned
  by the information in the \( t \) cells has dimension \( t \).

The rest of this paper is organized as follows. In Section \( \text{2} \) we present a simple upper
bound on the value of \( g(s) \). Though this bound is attained, we prove that \( g(s, t) < g(s) \)
for any fixed values of \( s \) and \( t \). We will also derive a more complex upper bound on \( g(s, t) \)
for various pairs \((s, t)\), and it will be shown to be attainable for \( 1 < s \leq 2 \). In Section \( \text{3} \)
we present a range of explicit constructions, mainly using set systems. We will prove that
the rates of most of these constructions approach a corresponding upper bound as \( t \) grows.
These constructions lead to interesting questions related to the set systems that are used.
We will also discuss how to use these constructions to obtain good bounds on the rate for any given pair \((s, t)\). We provide a conclusion in Section 4 where problems for future research are presented.

## 2 Upper Bounds

In this section we will be concerned first with a simple general upper bound (Theorem 3) on the rate of a \(k\)-PIR array code for a fixed value of \(s\) with \(s > 1\). This bound cannot be attained, but is asymptotically optimal (as \(t \to \infty\)). This will motivate us to give a stronger upper bound (Theorem 4) on the rate \(g(s, t)\) of a \([t \times s]k\)-PIR array code for various values of \(t\) that can sometimes be attained.

**Theorem 3.** For each rational number \(s > 1\) we have that \(g(s) \leq (s + 1)/(2s)\). There is no \(t\) such that \(g(s, t) = (s + 1)/(2s)\).

**Proof.** Suppose we have a \([t \times m, p]\) \(k\)-PIR array code with \(p/t = s\). To prove the theorem, it is sufficient to show that \(k/m < (s + 1)/2s\). Recall that we are assuming, without loss of generality, that if \(x_i\) can be derived from information on a certain server, then the singleton \(x_i\) is stored as the value of one of the cells of this server.

Let \(\alpha_i\) be the number of servers which hold the singleton \(x_i\) in one of their cells. Since each server has \(t\) cells, we find that \(\sum_{i=1}^{p} \alpha_i \leq tm\), and so the average value of the integers \(\alpha_i\) is \(tm/p = m/s\). So there exists \(u \in \{1, 2, \ldots, p\}\) such that \(\alpha_u \leq m/s\) (and we can only have \(\alpha_u = m/s\) when \(\alpha_i = m/s\) for all \(i \in \{1, 2, \ldots, p\}\)). Let \(S^{(1)}, S^{(2)}, \ldots, S^{(k)} \subseteq \{1, 2, \ldots, m\}\) be disjoint sets of servers, chosen so the span of the cells in each subset of servers contains \(x_u\). Such subsets exist, by the definition of a \(k\)-PIR array code. If no server in a subset \(S^{(j)}\) contains the singleton \(x_u\), the subset \(S^{(j)}\) must contain at least two elements (because we are assuming, without loss of generality, that if \(x_i\) can be derived from information on a certain server, then the singleton \(x_i\) is stored as the value of one of the cells of this server). So at most \(\alpha_u\) of the subsets \(S^{(j)}\) are of cardinality 1. In particular, this implies that \(k \leq \alpha_u + (m - \alpha_u)/2\).

Hence

\[
\frac{k}{m} \leq \frac{\alpha_u + (m - \alpha_u)/2}{m} = \frac{1}{2} + \frac{\alpha_u}{2m} \leq \frac{1}{2} + \frac{m/s}{2m} = \frac{1}{2} + \frac{1}{2s} = \frac{s + 1}{2s}.
\]

We can only have equality in (1) when \(\alpha_i = m/s\) for all \(i \in \{1, 2, \ldots, p\}\), which implies that all cells in every server are singletons. But then the span of subset of servers contains \(x_i\) if and only if it contains server with a cell \(x_i\), and so \(k \leq \alpha_i = m/s\). But this implies that the rate \(k/m\) of the array code is at most \(1/s = 2/2s\). This contradicts the assumption that the rate of the array code is \(k/m = (s+1)/2s\), since \(s > 1\). So \(k/m < (s+1)/2s\), as required. \(\square\)

**Theorem 4.** For any integer \(t \geq 2\) and any positive integer \(d\), we have

\[
g(1 + \frac{d}{t}, t) \leq \frac{(2d + 1)t + d^2}{(t + d)(2d + 1)} = 1 - \frac{d^2 + d}{(t + d)(2d + 1)}.
\]

**Proof.** Suppose we have a \([t \times m, p]\) \(k\)-PIR array code with \(p = t + d\). We aim to provide an upper bound on the rate \(k/m\) of this code.
For each \( i \in \{1, 2, \ldots, p\} \), let \( S^{(1)}_i, \ldots, S^{(k_i)}_i \subseteq \{1, 2, \ldots, m\} \) be disjoint sets of servers, chosen so that the cells in each subset of servers span a subspace containing \( x_i \). We choose these subsets so that \( k_i \) is as large as possible subject to this condition; so \( k = \min\{k_1, k_2, \ldots, k_p\} \leq (\sum_{i=1}^p k_i)/p = (\sum_{i=1}^p k_i)/(t + d) \). To prove the theorem, which asks for an upper bound on \( k/m \), it suffices to show that

\[
\sum_{i=1}^p k_i \leq \frac{(2d + 1)t + d^2}{2d + 1} m.
\]

Without loss of generality, we may assume that when server \( j \) contains a singleton entry \( x_i \) then \( \{j\} \) is one of the subsets \( S^{(1)}_i, \ldots, S^{(k_i)}_i \).

We say that a server is \textit{singleton} if all its cells are singletons; otherwise we say that a server is non-singleton. Let \( \ell \) be the number of singleton servers, and let \( r \) be the number of non-singleton servers. So \( \ell + r = m \).

For \( i \in \{1, 2, \ldots, p\} \), let \( \ell_i \) be the number of singleton servers with a cell equal to \( x_i \), and let \( r_i \) be the number of non-singleton servers with a cell equal to \( x_i \). Since every singleton server contains \( t \) distinct singleton cells, and every non-singleton server contains at most \( t - 1 \) singleton cells, we see that \( \sum_{i=1}^p \ell_i = t\ell \) and \( \sum_{i=1}^p r_i \leq (t - 1)r \).

Let \( f_i \) be the number of sets in the list \( S^{(1)}_i, \ldots, S^{(k_i)}_i \) that are of cardinality 2 or more, but contain at least one singleton server. None of the sets counted by \( f_i \) contain a server with a cell \( x_i \), since the sets have cardinality at least 2. Hence \( f_i \leq \ell - \ell_i \). Every set counted by \( f_i \) must involve a non-singleton server, as cells of the form \( x_u \) for \( u \neq i \) can never span a space containing \( x_i \). Moreover, the non-singleton servers involved cannot contain \( x_i \) as an entry, so \( f_i \leq r - r_i \).

When \( i \in \{1, 2, \ldots, p\} \) is fixed, there are exactly \( \ell_i + r_i \) sets \( S^{(j)}_i \) of size 1, and (by definition) there are \( f_i \) sets \( S^{(j)}_i \) that involve singleton servers. Every remaining set of the form \( S^{(j)}_i \) must involve at least 2 non-singleton servers, and so there are at most \((r - r_i - f_i)/2\) sets that remain. Hence

\[
k_i \leq \ell_i + r_i + f_i + (r - r_i - f_i)/2 = \ell_i + r/2 + r_i/2 + f_i/2.
\]

Since \( f_i \leq r - r_i \), we see that (2) implies

\[
k_i \leq \ell_i + r.
\]

Moreover, since \( f_i \leq \ell - \ell_i \), we see that (2) implies

\[
k_i \leq (\ell + \ell_i)/2 + (r + r_i)/2 = m/2 + \ell_i/2 + r_i/2.
\]

Our proof now splits into two cases. First suppose that \( r \leq dm/(2d + 1) \). The bound (3) implies that

\[
\sum_{i=1}^p k_i \leq \sum_{i=1}^p \ell_i + pr = t\ell + pr = t\ell + (t + d)r = tm + dr.
\]

The right hand side is maximised when \( r \) is as large as possible, in other words when \( r = dm/(2d + 1) \), and so

\[
\sum_{i=1}^p k_i \leq tm + d^2m/(2d + 1) = \frac{t(2d + 1) + d^2}{2d + 1} m,
\]
as required.

Now suppose that \( r \geq \frac{dm}{2d+1} \). Then (4) implies that

\[
\sum_{i=1}^{p} k_i \leq \frac{pm}{2} + \sum_{i=1}^{p} \ell_i/2 + \sum_{i=1}^{p} r_i/2 \\
\leq \frac{pm}{2} + t\ell/2 + (t-1)r/2 \\
= \left(\frac{(p+t)m-r}{2}\right).
\]

The right hand side is maximised when \( r \) is as small as possible, in other words when \( r = \frac{dm}{2d+1} \). So, since \( p = d + t \),

\[
\sum_{i=1}^{p} k_i \leq \frac{(p+t)m}{2} - \frac{(d/(2d+1))m}{2} = \frac{(d + 2t)(2d+1) - d}{2(2d+1)} m = \frac{t(2d+1) + d^2}{2d+1} m,
\]

as required. \( \Box \)

3 Constructions

In this section we will propose various constructions for PIR array codes; these yield lower bounds on \( g(s, t) \) and on \( g(s) \). The constructions are based on various set systems: the set systems consist of subsets \( X \subseteq \{1, 2, \ldots, p\} \), which correspond to one of the \( t \) linear combinations \( \sum_{i \in X} x_i \) stored in a cell of a server. Some of our constructions choose the subsets \( X \) based on some block design.

To illustrate the approach, we begin by rephrasing the example of a PIR array code given in [13]. The example covers the case where \( s \) is an integer such that \( s = t + 1 > 2 \) and \( p = st = t(t+1) \).

In the PIR array code from Fazeli et al, we have two types of servers. Each server of Type A stores only singletons. This is done in all possible ways, so we have \( \binom{p}{t} \) servers of Type A. There are \( t\binom{p}{t} / (t+1) \) servers of Type B, indexed by the elements of a set \( B \) which is chosen as follows. Let \( B \) be a set of \( t\binom{p}{t} / (t+1) \) (unordered) partitions \( X_1 \cup X_2 \cup \cdots \cup X_t \) of \( \{1, 2, \ldots, p\} \) into \( t \) sets, each of size \( t+1 \). This set is chosen so that every \( (t+1) \)-subset occurs exactly once in a partition; \( B \) exists by Baranyai’s Theorem [2]. A server of type B consists of cells which are sums of \( t+1 \) parts: cell \( j \) of the server corresponding to \( X_1 \cup X_2 \cup \cdots \cup X_t \) stores the linear combination \( \sum_{i \in X_j} x_i \).

We can reconstruct \( x_i \) from any of the servers of Type A that contain \( x_i \) as a singleton. If a server of Type A does not contain \( x_i \), we pair it with the unique server of Type B that contains a cell equal to the sum of \( x_i \) and the cells in Type A server; this pair of servers can reconstruct \( x_i \). Note that we do not pair a server of Type B with a more than one server of Type A. Thus the array code has the \( k \)-PIR property with \( k = \binom{p}{t} \).

There are \( m = \binom{p}{t} + t\binom{p}{t} / (t+1) \) servers, and so we have PIR rate equal to \( k/m = (t+1)/(2t+1) \). Thus \( g(t+1, t) = g(s, s-1) \geq (t+1)/(2t+1) = s/(2s-1) \) whenever \( s = t+1 > 2 \), and so for any integer \( s > 2 \) we have \( g(s) \geq s/(2s-1) \).

The constructions presented in the following subsections yield an improvement on the lower bound above. They also cover all rational values of \( s > 1 \), and not just integer values

7
of $s$. We are interested in constructions with the number of servers as small as possible. This goal will be achieved using Hall’s marriage Theorem:

**Theorem 5.** In a finite bipartite graph $G = (V_1 \cup V_2, E)$, there is perfect matching if for each subset $X$ of $V_1$, the number of vertices in $V_2$ connected to vertices of $X$ has at least size $|X|$.

**Corollary 1.** A finite regular bipartite graph has a perfect matching.

### 3.1 A shortening construction

In this subsection we provide a shortening construction, reducing the number of parts by one. The construction can be applied on any given PIR array code to provide a new code with a new set of parameters. It can be applied repeatedly, giving a sequence of codes of the same rate. But note that the optimal rate (for given parameters) could well rise as the shortening is repeatedly applied, making the array code less efficient in terms of rate. The construction is particularly useful when the parameters of the PIR array code are small.

Suppose we have a $k$-PIR array code having parameters $p$, $t$, $s = p/t$, $k$, and $m$. We choose a part $x_i$, and remove it from all linear combinations in the array code; if the cell is the singleton $x_i$ we replace this cell with a linear combination that is independent from the remaining cells of the server. We obtain a $k$-PIR array code with parameters $p'$, $t'$, $m'$ and $s'$ where $p' = p - 1$, $t' = t$, $m' = m$ and $s' = p'/t' = s - 1/t$. In particular, the PIR rates of the codes are the same $k/m$. We summarize the result of this construction with the following theorem.

**Theorem 6.** For any integer $t > 1$ and any rational number $s > 1 + 1/t$, we have $g(s, t) \leq g(s - 1/t, t)$.

### 3.2 A construction for $s = 2$

The only integer value which is not covered by the PIR array codes in [13] is $s = 2$. In this subsection we present a simple optimal construction for $s = 2$ which demonstrates the ideas in our constructions which follow.

**Construction 1.** ($s = 2$ and $p = 2t$, where $t \geq 2$)

There are two types of server. Servers of Type A have $t$ singleton cells. This is done in all possible ways, so there are $\binom{2t}{t}$ servers of Type A. Each server of Type B has $t - 1$ singleton cells; its remaining cell stores the sum of the $t + 1$ parts not in the singletons. Again, this is done in all possible ways, so there are $\binom{2t-1}{t-1}$ servers of Type B.

Construction 1 implies the following result.

**Theorem 7.** If $t \geq 2$, then $g(2, t) \geq (3t + 1)/(4t + 2)$.

**Proof.** Consider Construction 1. The total number of servers is $m = \binom{2t}{t} + \binom{2t-1}{t-1} = \binom{2t+1}{t}$. Each part $x_i$ is stored as a singleton cell in $\binom{2t-1}{t-1}$ servers of Type A, and so can be recovered by that server acting alone. There are $\binom{2t-1}{t}$ servers of Type A that remain. Given a server...
of Type B, either \( x_i \) is either stored as a singleton in one of the cells of the server, or it is stored as a linear combination involving \( t \) other parts. In the former case, the server can recover \( x_i \) acting alone. In the latter case, the server can recover \( x_i \) when paired with the unique server of Type A that stores the \( t \) other parts as singletons. Note that this Type A server does not store \( x_i \) as a singleton and so has not been used beforehand. Since there are \( \binom{2t-1}{t-1} \) Type A servers that contain \( x_i \) as a singleton, and there are \( \binom{2t-1}{t} \) servers of Type B, we see that \( k = \binom{2t-1}{t-1} + \binom{2t-1}{t} \). A simple computation implies that \( k/m = (3t+1)/(4t+2) \) and hence \( g(2,t) \geq (3t+1)/(4t+2) \).

Theorems 4 and 7 together imply the following.

**Corollary 2.**

(i) If \( t \geq 2 \), then \( g(2,t) = (3t+1)/(4t+2) \).

(ii) \( g(2) = 3/4 \).

### 3.3 Constructions for \( 1 < s < 2 \)

In this subsection we present constructions for PIR array codes when \( s \) is a rational number between 1 and 2. The construction will be generalized in Subsection 3.4 when \( s \) any rational number greater than 1, but the special case considered here deserves separate attention for three reasons: it is simpler than its generalization; the constructed PIR array code attains the bound of Theorem 4 while we do not have a proof of similar result for the generalization; and finally the analysis of the generalization is slightly different.

**Construction 2.** \((s = 1 + d/t \) and \( p = t + d \) for \( t > 1 \), \( d \) a positive integer, \( 1 \leq d \leq t - 1 \)).

Let \( \vartheta \) be the least common multiple of \( d \) and \( t \). There are two types of servers. Servers of Type A store \( t \) singletons. Each possible \( t \)-subset of parts occurs \( \vartheta/d \) times as the set of singleton cells of a server, so there are \( \binom{p}{t} \vartheta/d \) servers of Type A. Each server of Type B has \( t - 1 \) singleton cells in \( t - 1 \) cells; the remaining cell stores the sum of the remaining \( p - (t - 1) = d + 1 \) parts. Each possible \((t - 1)\)-set of singletons occurs \( \vartheta/t \) times, so there are \( \binom{p}{t-1} \vartheta/t \) servers of Type B.

**Theorem 8.** For any given \( t > 1 \) and \( 1 \leq d \leq t - 1 \),

\[
g(1 + d/t, t) \geq \frac{(2d+1)t+d^2}{(t+d)(2d+1)}.
\]

**Proof.** The total number of servers in Construction 2 is \( m = \binom{t+d}{t} \vartheta/d + \binom{t+d}{d+1} \vartheta/t \). We now calculate \( k \) such that Construction 2 has the \( k \)-PIR property. To do this, we compute for each \( i, 1 \leq i \leq p \), a collection of pairwise disjoint sets of servers, each of which can recover the part \( x_i \).

There are \( \binom{t+d-1}{t-1} \vartheta/d \) servers of Type A containing \( x_i \) as a singleton cell. Let \( V_1 \) be the set of \( \binom{t+d-1}{t-1} \vartheta/d \) remaining servers of Type A. There are \( \binom{t+d-1}{t-2} \vartheta/t \) servers of Type B containing \( x_i \) as a singleton cell. Let \( V_2 \) be the set of \( \binom{t+d-1}{t-2} \vartheta/t \) remaining servers of Type B.

We define a bipartite graph \( G = (V_1 \cup V_2, E) \) as follows. Let \( v_1 \in V_1 \) and \( v_2 \in V_2 \). Let \( X_1 \subseteq \{x_1, x_2, \ldots, x_p\} \) be the set of \( t \) singleton cells of the server \( v_1 \). Let \( X_2 \subseteq \{x_1, x_2, \ldots, x_p\} \)
be the parts involved in the non-singleton cell of the server \( v_2 \). (So \( X_2 \) is the set of \( d + 1 \) parts that are not singleton cells of \( v_2 \). Note that \( x_i \in X_2 \).) We draw an edge from \( v_1 \) to \( v_2 \) exactly when \( X_2 \setminus \{ x_i \} \subseteq X_1 \). Note that \( v_1 \) and \( v_2 \) are joined by an edge if and only if the servers \( v_1 \) and \( v_2 \) can together recover \( x_i \).

The degrees of the vertices in \( V_1 \) are all equal; the same is true for the vertices in \( V_2 \). Moreover, \(|V_1| = \binom{t+d-1}{t-1} \vartheta/d = \binom{t+d-1}{t-2} \vartheta/t = |V_2|\). So \( G \) is a regular graph. So by Corollary \( \text{1} \) there exists a perfect matching in \( G \). The edges of this matching form \(|V_1|\) disjoint sets of servers, each of which can recover \( x_i \). Thus, we have that \( k = \binom{t+d-1}{t-1} \vartheta/d + \binom{t+d-1}{t-2} \vartheta/t + \binom{t+d-1}{t} \vartheta/d = m - \binom{t+d-1}{t} \vartheta/d \).

Finally, some simple algebraic manipulation shows us that

\[
g(1 + d/t, t) \geq \frac{k}{m} = \frac{(2d + 1)t + d^2}{(t + d)(2d + 1)} \ .
\]

Combining Theorems \( \text{4} \) and \( \text{8} \) we find the following.

**Corollary 3.**

(i) For any given \( t \) and \( d \), when \( s = 1 + d/t \) we have

\[
g(s, t) = 1 - \frac{d^2 + d}{(t + d)(2d + 1)} = \frac{t}{t + d} + \frac{d^2}{(t + d)(2d + 1)} = \frac{s + 1 + 1/d}{(2 + 1/d)s} \ .
\]

(ii) For any rational number \( 1 < s < 2 \), we have \( g(s) = (s + 1)/(2s) \).

**Proof.** Theorems \( \text{4} \) and Theorem \( \text{8} \) directly establish the first two equalities in Part (i) of the corollary. The final equality in Part (i) follows from the substitution \( s = 1 + d/t = (d + t)/t \):

\[
g(s, t) = \frac{1}{s} + \frac{d^2}{st(2d + 1)} = \frac{1}{s} + \frac{d}{s(2 + \frac{d}{s})} = \frac{1}{s} + \frac{s - 1}{s(2 + \frac{d}{s})} = \frac{s + 1 + 1/d}{(2 + 1/d)s} \ .
\]

As \( t \) and \( d \) tend to infinity with \( d/t \) fixed, the value of \( s \) does not change but we see that \( g(s, t) \to (s + 1)/(2s) \). This establishes the final statement of the corollary.

Corollary \( \text{8} \) shows that Construction \( \text{2} \) has optimal PIR rate. But the number of servers used for this code is large. PIR array codes with a smaller number of servers are more interesting for applications. The PIR array codes presented in the remainder of this subsection have lower rates than the rates of the codes produced by Construction \( \text{2} \) but their rates are asymptotically optimal and the number of servers they use is significantly smaller.

The servers in the following construction are similar to those of Type B in Construction \( \text{1} \), although the construction itself is different. It will be generalized, in some sense, in Subsection 3.4, while Construction \( \text{1} \) will be generalized similarly in Subsection 3.5.

**Construction 3.** \( (s = 1 + d/t \) and \( p = t + d \), where \( t > 1 \) and \( d \) is a positive integer such that \( 1 \leq d \leq t - 1 \))

All servers have the same type in this construction. Each server has \( t - 1 \) singleton parts; the remaining cell is the sum of the \( d + 1 \) parts that do not occur as singletons. Each subset of \( t - 1 \) singleton parts occurs in two servers.
Theorem 9. Construction 3 yields a PIR array code with PIR rate $1 - (d + 1)/(2(t + d))$ and $2^{(t+d)\choose (t-1)}$ servers.

Proof. The total number of servers in Construction 3 is clearly $m = 2^{(t+1)\choose (t-1)} = 2^{(t+d)\choose (t-1)}$.

Fix a part $x_i$. There are $2^{(t+d-1)\choose (d+1)}$ servers in which $x_i$ is singleton. These servers can all individually compute the part $x_i$. There are $2^{(t+d-1)\choose d}$ servers that remain, in which $x_i$ is not a singleton. We divide these remaining servers into two sets $V_1$ and $V_2$ of size $2^{(t+d-1)\choose (d+1)}$, where we place the two servers whose cells contain the same information in different sets. We define a bipartite graph $G = (V_1 \cup V_2, E)$ by joining $v_1 \in V_1$ and $v_2 \in V_2$ by an edge as follows. Let $X_1 \subseteq \{x_1, x_2, \ldots, x_p\}$ be the set of $d+1$ parts that do not appear in singleton cells in $v_1$. Let $X_2$ be the corresponding set associated with $v_2$. We join $v_1$ and $v_2$ by an edge if and only if $X_1$ and $X_2$ are disjoint. Note that when $v_1$ and $v_2$ are joined by an edge, the information in $v_1$ and $v_2$ together can be used to compute $x_i$.

The graph $G$ is clearly regular, and so Corollary 1 implies there exists a perfect matching in $G$. The edges of this matching give rise to disjoint pairs of servers, and each pair can recover $x_i$. Thus, taking into account those servers having $x_i$ as a singleton cell, we may take $k = 2^{(t+d-1)\choose (d+1)} + 2^{(t+d-1)\choose d}$. This implies that $k/m = 1 - (d + 1)/(2(t + d))$ and the claim follows.

The lower bound on $k/m$ implied from Construction 3 can be improved by applying the shortening construction on the PIR array codes obtained in Construction 4. It will imply that $k/m = (3t+1)/(4t+2)$, using a small number of servers, which is better than the bound implied by Construction 3.

The next pair of constructions cover values of $s$ which are slightly larger than one.

Construction 4. ($s = 1 + 1/t$, $p = t + 1$, where $t$ is odd)

There are two types of servers. There are $t + 1$ servers of Type A, with $t$ singleton cells. (So exactly one part is not stored in each Type A server.) There are $(t + 1)/2$ servers of Type B. The $j$th server of Type B stores the sum $x_{2j-1} + x_{2j}$ in one cell, and the remaining $t − 1$ parts (those not equal to $x_{2j-1}$ or $x_{2j}$) as singleton cells.

To recover $x_i$ in Construction 4 we see that there are there are $t$ servers of Type A which store $x_i$ in one of their cells as a singleton, and $(t − 1)/2$ servers of Type B. The only server of Type B that does not store $x_i$ as a singleton stores either $x_{i-1} + x_i$ or $x_i + x_{i+1}$. But this server can be paired with the server of Type A that does not store $x_i$: the Type A server stores $x_{i-1}$ or $x_{i+1}$ as appropriate, so this pair of servers can together recover $x_i$. Thus, in this case $m = t + 1 + (t + 1)/2 = (3t + 3)/2$, and $k = (3t + 1)/2$. Thus $k/m = (3t + 1)/(3t + 3)$.

Construction 5. ($s = 1 + 1/t$, $p = t + 1$, where $t$ is even)

There are two types of servers. There are $2(t + 1)$ servers of Type A, each storing $t$ singletons, with one part not stored in each server. Each part fails to be stored on exactly two servers of Type A. There are $t + 1$ servers of Type B, where the $j$th server stores $x_j + x_{j+1}$ (subscripts taken modulo $t + 1$) in one cell, and the remaining $t − 1$ parts as singleton cells.

To reconstruct $x_i$ using the PIR array code of Construction 5 we first note that there are $2t$ servers of Type A and $t − 1$ servers of Type B which store $x_i$ as a singleton cell. The two servers in $B$ which do not store $x_i$, store either $x_{i-1} + x_i$ or $x_i + x_{i+1}$: they can each be
paired with one of the two servers of Type A that does not store $x_i$ as a singleton, so both pairs can compute $x_i$. Hence, we can take $k = 2t + (t - 1) + 2 = 3t + 1$. Since $m = 3t + 3$, we find that $k/m = (3t + 1)/(3t + 3)$.

To summarise, the rates of the PIR array codes of Constructions 4 and 5 attain the upper bound of Theorem 4 with a small number of servers. (In fact, it can be proved that these constructions use the smallest possible number of servers.)

We will now give the idea how to generalize Construction 4 without a formal proof. Recall that an $S(a, b, c)$ Steiner system is a set $S$ of $b$-subsets (blocks) of a $c$-set (of points), with the property that every set of $a$ points is contained in exactly one block.

Construction 6. ($s = 1 + d/t$, $p = t + d$, and there exists a Steiner system $S(d, d + 1, p)$)

Let $S$ be a $S(d, d + 1, p)$ Steiner system on the set of points $\{1, 2, \ldots, p\}$. We define servers of two types. There are $\binom{t+d}{t} = \binom{t+d}{d}$ servers of Type A: each server stores a different subset of parts in $t$ singleton cells. There are $\frac{d}{d+1} \binom{t+d}{d}$ servers of Type B, indexed by a set that repeats each of the $\frac{1}{d+1} \binom{t+d}{d}$ blocks $B \in S$ a total of $d$ times. One cell in a server of Type B contains the sum $\sum_{i \in B} x_i$; the remaining $t - 1$ cells contain the $t - 1$ parts not involved in this sum.

The number of servers in the PIR code obtained from Construction 6 is $m = \binom{t+d}{d} + \frac{d}{d+1} \binom{t+d}{d}$. Just as in constructions above, we can prove that we may take $k = m - \binom{t+d-1}{d-1}$ in this construction, by first counting the servers of either type that contain $x_i$ is a singleton, and then pairing up the servers of Type B with servers of Type A using a matching obtained by a regular bipartite graph. The rate of the code is

$$\frac{k}{m} = 1 - \frac{d(d+1)}{(t+d)(2d+1)}$$

which attains the bound of Theorem 4. Unfortunately, the number of known Steiner systems $S(d, d + 1, t + d)$ is limited: though infinitely many of them are known to exist for all values of $d$, the proof is not constructive [20].

This last construction can be modified by using only servers of Type B. We define $m = \frac{2}{d+1} \binom{t+d}{d}$, and index the servers using each block $B$ in the Steiner system twice. We pair those servers which do not have $x_i$ as a singleton cell using a matching obtained by a regular bipartite graph (each side of the graph has $\binom{t+d-1}{d-1}/d$ vertices (servers), where the two servers associated with a block are in different parts of the vertex bipartition). One can verify that $k = m - \binom{t+d-1}{d-1}/d$ and that the rate of the code is

$$\frac{k}{m} = 1 - \frac{d+1}{2(t+d)}.$$  

This is slightly lower than the upper bound of Theorem 4, but the construction uses a smaller number of servers. We can further reduce the number of servers considerably, maintaining the same PIR rate, if we take sets of singletons of server of Type B to be the elements in the blocks of a Steiner system $S(d, t - 1, p)$.

Finally, the number of servers of Type A can be reduced by restricting the sets of singletons in such a server to lie in a Steiner system $S(a, t, p)$, or a block design $S_\lambda(a, t, p)$. This will reduce the number of servers considerably without affecting the PIR rate. The complete analysis will appear in the final version of this paper.
3.4 Constructions when $s$ is not an integer

In this subsection we are concerned in the largest range of numbers for which we don’t know the exact value of the asymptotic rate of PIR array codes. In this section, $s$ will be an arbitrary rational number with $s > 2$ and $s$ not an integer. (Constructions when $s$ is an integer will be given in the following section.)

The first construction is a generalization, in some sense, of Construction 3.

**Construction 7.** $(s = (rt - (r - 2)r + 1)/t, p = rt - (r - 2)r - 1, \text{ where } 3 \leq r \leq t)$

All the servers have the same type in this construction. Each server has $t - r + 1$ singleton cells. The remaining $r - 1$ cells of any server contain sums of $t - r + 2$ parts from the remaining $p - t + r - 1$ parts; the terms in these sums are pairwise disjoint. Each possibility for the set of singleton cells and the partition of the remaining cells into disjoint sums in this manner occurs exactly once.

**Theorem 10.** If $3 \leq r \leq t$ then Construction 7 implies that

$$g\left(r - \frac{(r - 2)r + 1}{t}, t\right) \geq \frac{1}{2} + \frac{t - r + 1}{2(rt - (r - 2)r - 1)}.$$

**Proof.** The number $m$ of servers in the construction is

$$m = \frac{(rt - (r - 2)r - 1)!}{(t - r + 1)!(r - 1)!(t - r + 2)!^{r - 1}}.$$

A part $x_i$ is contained in a singleton cell in

$$\frac{(rt - (r - 2)r - 2)!}{(t - r)!(r - 1)!(t - r + 2)!^{r - 1}}$$

servers. There are

$$\frac{(rt - (r - 2)r - 2)!}{(t - r + 1)!^2(r - 2)!(t - r + 2)!^{r - 2}}$$

servers that remain. Each of these servers is associated with two disjoint $(t - r + 1)$-subsets of $\{x_1, x_2, \ldots, x_p\} \setminus \{x_i\}$: the parts occurring in the singleton cells of $S$, and the parts that appear in the sum in the cell involving $x_i$. We can pair up the two servers where these two subsets are swapped and all remaining sums are the same; this pair of servers can recover $x_i$. Therefore we may take

$$k = \frac{(rt - (r - 2)r - 2)!}{(t - r)!(r - 1)!(t - r + 2)!^{r - 1}} + \frac{(rt - (r - 2)r - 2)!}{(t - r + 1)!^2(r - 2)!(t - r + 2)!^{r - 2}}/2.$$

A straightforward algebraic computation implies that

$$g\left(r - \frac{(r - 2)r + 1}{t}, t\right) \geq \frac{1}{2} + \frac{t - r + 1}{2(rt - (r - 2)r - 1)}.$$  \hfill \square

Construction 7 and Theorem 10 imply the following result.
Corollary 4. If $3 \leq r \leq t$ and $s = r - ((r - 2)r + 1)/t$, then

$$g(s,t) \geq \frac{s+1}{2s} - \frac{r-1}{2st}.$$ 

Proof. Substituting $s = r - ((r - 2)r + 1)/t$ in the inequality

$$g \left( r - \frac{(r - 2)r + 1}{t}, t \right) \geq \frac{1}{2} + \frac{t - r + 1}{2(rt - (r - 2)r - 1)}$$

yields

$$g(s,t) \geq \frac{1}{2} + \frac{t - r + 1}{2st} = \frac{s+1}{2s} - \frac{r-1}{2st}.$$ 

Now, it is obvious that if $t$ is large then the ratio $k/m$ obtained in Construction 7 becomes close to the bound of Theorem 3. However, because of the interdependency of $t$, $s$ and $r$ we never attain this bound asymptotically for a fixed value of $s$. To overcome this problem with Construction 7 we will next present a construction which has asymptotically optimal rate. The construction can be used for all rational numbers $s$ with $s > 2$. We believe the construction produces PIR array codes with the largest possible rate; it is a generalization of Constructions 1 and 2. However the construction has the disadvantage that it requires many more servers.

Construction 8. $(s = r + d/t, p = rt + d)$, where $r \geq 2$ is an integer, $t \geq r$, $1 \leq d \leq t - 1$)

There are two types of servers. Each server of Type A stores $t$ singleton parts. Every possible combination of singleton parts occurs $n_A$ times, where

$$n_A = \frac{(rt + d - t - 1)!(rt + d - t + r)}{(t - r)!(d + 1)!(r - 2)!(t + 1)^{r-2}}.$$ 

(Note that $n_A$ is an integer.) A server of Type B contains $t - r$ singleton cells, one cell containing a sum of $d + 1$ parts and $r - 1$ cells in which a linear combination of $t + 1$ parts is stored. The sets of terms in the sums are pairwise disjoint, and are also disjoint from the singletons. Each possible combination of singletons and sums of this form occurs in exactly $n_B$ servers of Type B, where $n_B = (r - 1)(t + 1)$.

Theorem 11. For any given integers $r > 1$ and $r \leq t$, and $1 \leq d \leq t - 1$,

$$g(r + d/t, t) \geq 1 - \frac{(rt + d - t + r)(rt + d - t)}{(rt + d)(2rt + 2d - 2t + r)}.$$ 

Proof. The total number of servers in Construction 8 is

$$m = \binom{rt + d}{t} n_A + \frac{(rt + d)!}{(t - r)!(d + 1)!(r - 1)!(t + 1)^{r-1} n_B}.$$

The number of servers of Type A which do not contain $x_i$ as a singleton is $\binom{rt + d - 1}{t - 1} n_A$ and the number of servers of Type B which do not contain $x_i$ as a singleton is $\frac{(rt + d - t + r)!}{(d + 1)!(r - 1)!(t + 1)^{r-1} n_B}$. 

14
From the remaining servers (which do not contain \( x_i \) as a singleton) we construct a bipartite graph \( G = (V_1 \cup V_2, E) \): the vertices of \( V_1 \) and \( V_2 \) are those of Type A and Type B respectively. Since

\[
\binom{rt + d - 1}{t} \binom{rt + d - t - 1}{(t - r)} (rt + d)(d+1)(r-2)!(t+1)!^{r-2} = \binom{rt + d - t + r}{t - r} \binom{rt + d - t + r}{t + 1} (d+1)! (r-1)! (t+1)!^{r-1},
\]

it follows that \( |V_1| = |V_2| \). We define edges \( E \) so that the servers at the ends of an edge can reconstruct \( x_i \). So \( \{v_1, v_2\} \) is an edge in \( E \) if and only either:

- there is a \( t \)-subset \( Y \) of parts that is the set of singleton parts of \( v_2 \), and such that a cell of \( v_1 \) contains the sum \( \sum_{x_j \in Y \cup \{x_i\}} x_j \); or
- there is a \( d \)-subset \( Z \) of parts that is contained in the set of singleton parts of \( v_1 \), and such that a cell in \( v_2 \) contains the sum \( \sum_{x_j \in Z \cup \{x_i\}} x_j \).

It is not hard to show that this bipartite graph is regular and so, by Theorem \([\star]\) there exists a perfect matching in \( G \). The ends of the edges in this matching form disjoint pairs of servers; which \( x_i \) can be computed from each pair. This analysis implies that

\[
k = m - \binom{rt + d - 1}{t} \binom{rt + d - t - 1}{(t - r)} (rt + d)(d+1)(r-2)!(t+1)!^{r-2}.
\]

One can verify that

\[
m = \frac{(rt + d)!}{(t - r)! (d+1)! (r-2)! (t+1)!^{r-2}} \quad \text{and hence} \quad g(r + d/t, t) \geq \frac{k}{m} = 1 - \frac{(rt + d - t + r)(rt + d - t)}{(rt + d)(2rt + 2d - 2t + r)}.
\]

\[\square\]

**Corollary 5.** If \( s > 2 \) is a rational number which is not an integer, then \( g(s) = (s+1)/(2s) \).

**Proof.** This is an immediate consequence of Theorem \([\|]\). For when \( s = r + d/t \), \( r \geq 2 \), \( t \geq 2 \), and \( 1 \leq d \leq t - 1 \) then

\[
g(s, t) \geq 1 - \frac{(rt + d - t + r)(rt + d - t)}{(rt + d)(2rt + 2d - 2t + r)}.
\]

For a fixed value of \( r \) is fixed, we let \( d \) and \( t \) tend to infinity with the ratio \( d/t \) fixed. Then

\[
g(s) = 1 - \frac{(rt + d - t)(rt + d - t)}{(rt + d)(2rt + 2d - 2t)} = \frac{1}{2} + \frac{t}{2(rt + d)} = \frac{1}{2} + \frac{1}{2(r + \frac{d}{t})} = \frac{1}{2} + \frac{1}{2s} = \frac{s + 1}{2s}.
\]

\[\square\]

We believe that the rate of the code obtained in Construction \([\|] \) is optimal. There is a construction with a slightly lower rate that uses fewer servers when \( t - r \geq d \): pairing servers of Type B that recover \( x_i \) from the sum with \( d + 1 \) terms and \( t - r \) singleton cells. The number of servers in Type A will be reduced, but servers of Type A will still be needed for pairing to recover \( x_i \) from servers in Type B that contain \( x_i \) in a sum of \( t + 1 \) parts. We omit the details of this construction.

Finally, we note that the analysis we have does not work if \( s \) is an integer (namely when \( d = t \)), as the combinatorial computations will be slightly different. The related construction with its computations is given in the next subsection.
3.5 Constructions when $s$ is an integer, $s > 2$

It was discussed at the start of this section, the construction given in [13] for any integer $s > 2$ and $t = s − 1$ yields $g(s) ≥ s/(2s − 1)$. In this section we present two other ways to obtain a PIR array code for integer values of $s$. We start by considering an improved lower bound on the optimal rate, following from Construction 7.

In Construction 7 we start by picking an integer $r$, and continue by choosing an integer $t$ such that $t ≥ r$. For the pair $(r, t)$, we have that $s = r − ((r − 2)r + 1)/t$. If we want $s$ to be an integer, then $t$ must divide $(r − 2)r + 1$. A good way of ensuring this is by choosing $t = (r − 2)r + 1$, which implies that $s = r − 1$. The analysis after Construction 7 shows that

$$g(r − 1, (r − 2)r + 1) ≥ \frac{1}{2} + \frac{(r − 2)r + 1 − r + 1}{2(r((r − 2)r + 1) − (r − 2)r − 1)}.$$

By substituting $r = s + 1$ in this equation, we obtain

$$g(s, s^2) ≥ \frac{1}{2} + \frac{s − 1}{2} · s^2 = \frac{s + 1}{2s} − \frac{1}{s^2}.$$

This lower bound is very close to the upper bound of Theorem 3.

Next, we will present a construction for all integer values $s$ that generalises Construction 1.

**Construction 9.** $(s > 2$ an integer and $p = st$, where $t ≥ s)$

There are two types of servers. Each possible subset of $t$ parts is stored as singleton cells in

$$\frac{(ts − t − 1)!}{(t − s + 1)!(s − 2)!(t + 1)!^{s−2}}$$

servers of Type A. Each server of Type B contains $t − s + 1$ singleton cells, with the remaining $s − 1$ cells containing a sum of $t + 1$ parts. There is exactly one server of Type B for each way of doing this.

**Theorem 12.** For any given integers $s > 1$ and $t ≥ s$,

$$g(s, t) ≥ 1 − \frac{(s − 1)(t + 1)}{s(2t + 1)} = \frac{st + t + 1}{s(2t + 1)}.$$

**Proof.** The total number $m$ of servers in Construction 9 is

$$m = \binom{ts}{t} \frac{(ts − t − 1)!}{(t − s + 1)!(s − 2)!(t + 1)!^{s−2}} + \binom{ts}{t} \frac{(ts)!}{(t − s + 1)!(s − 1)!(t + 1)!^{s−1}}.$$

To compute $k$, we first note that there are exactly $\binom{s−1}{t−1}$ combinations of $t$ parts which contain $x_i$ and hence there are exactly

$$\binom{st−1}{t−1} \frac{(ts − t − 1)!}{(t − s + 1)!(s − 2)!(t + 1)!^{s−2}}$$

servers of Type A that store $x_i$ as a singleton. There are

$$\binom{ts−1}{t−s} \frac{(t+1)(s−1)!}{(s−1)!(t+1)!^{s−1}}$$
servers of Type B that store $x_i$ as a singleton. A pair of servers can recover $x_i$ if one is a Type B server storing a sum of $t + 1$ parts including $x_i$, and the other is a server of Type A that stores the other $t$ parts in the sum as singletons. Given a set $S$ of $t + 1$ parts that includes $x_i$, there are
\[
\frac{(ts - t - 1)!}{(t - s + 1)!(s - 2)!(t + 1)!^{s-2}}
\]
servers of Type B with a cell equal to $\sum_{x_j \in S} x_j$. But this is also the number of servers of Type A which store the $t$ parts $S \setminus \{x_j\}$ as singletons. Hence, $x_i$ can be recovered by
\[
\binom{ts - 1}{t} \frac{(ts - t - 1)!}{(t - s + 1)!(s - 2)!(t + 1)!^{s-2}}
\]
pairs of servers in this manner, since there are $\binom{ts - 1}{t}$ choices for $S$. Thus,
\[
k = \binom{st - 1}{t - 1} \frac{(ts - t - 1)!}{(t - s + 1)!(s - 2)!(t + 1)!^{s-2}} + \binom{ts - 1}{t - s} \frac{(ts - t - 1)!}{(s - 1)!(t + 1)!^{s-1}} + \binom{ts - 1}{t} \frac{(ts - t - 1)!}{(t - s + 1)!(s - 2)!(t + 1)!^{s-2}}
\]
\[
= m - \binom{ts - 1}{t} \frac{(ts - t - 1)!}{(t - s + 1)!(s - 2)!(t + 1)!^{s-2}}.
\]
After some algebraic manipulation we see that
\[
g(s, t) \geq \frac{k}{m} = 1 - \frac{(s - 1)(t + 1)}{s(2t + 1)} = \frac{st + t + 1}{s(2t + 1)}.
\]

Combining now Theorems 3 and 12 we have:

**Corollary 6.** For any given integer $s \geq 2$, $g(s) = (s + 1)/(2s)$.

We believe that the rate of the code obtained in Construction 9 is optimal. We may obtain a slightly lower rate using considerably fewer servers using only servers of Type B in a similar way to previous constructions.

**Construction 10.** ($s > 2$ an integer, and $p = st$ where $(s - 1)t = \ell b$, and $t \geq \ell + b$)

All the servers are of the same type in this construction. In each server there are $t - \ell$ singleton cells and $\ell$ cells summing $(b + 1)$ parts. Two servers store each possible partition of the $st$ parts into sums and singletons in their cells.

**Theorem 13.** The rate of the code generated in Construction 10 is
\[
\frac{k}{m} = \frac{s + 1}{2s} - \frac{\ell}{2st}.
\]
Proof. As the proof is similar to previous proofs, we will only sketch it. The number of servers is

\[ m = \frac{2(st)!}{(t-\ell)!(b+1)!^\ell}. \]

The number of servers which do not contain \( x_i \) in a singleton cell is

\[ \frac{2(st - 1)!}{(t-\ell)!b!(\ell - 1)!(b+1)!^{\ell-1}}. \]

We pair the remaining servers using a perfect matching in a bipartite graph. So

\[ k = m - \frac{(st - 1)!}{(t-\ell)!b!(\ell - 1)!(b+1)!^{\ell-1}}. \]

Thus, the rate is

\[ \frac{k}{m} = \frac{s + 1}{2s} - \frac{\ell}{2st}. \]

Clearly to get the best rate from Construction \[ \text{10} \] we must choose \( \ell \) to be as small as possible.

3.6 Analysis of the Constructions

In this subsection we will summarize and analyze the constructions from the previous subsections. We start with an immediate consequence of Corollaries \[ \text{2, 3, 5, and 6} \]:

**Theorem 14.** For any rational number \( s > 1 \), we have \( g(s) = \frac{s + 1}{2s} \).

Suppose we are required to design a PIR array code for a given application. One parameter that might well be fixed is the fraction of the file, \( \frac{1}{s} \), which is stored in a server, and the constructions producing the lower bounds on \( g(s) \) will provide codes of good rate. But many applications will also mandate a value for \( t \), the number of cells contained in each server. Amongst PIR array codes with fixed values of \( s \) and \( t \), examples with \( m \), the number of servers, as small as possible are to be preferred. For a large range of pairs \( (s, t) \) we have given constructions which we believe to be optimal in terms of rate. But the number of servers in our constructions might be higher than necessary. Moreover, we might prefer a construction with a slightly smaller rate, but a considerably smaller number of servers. We will now comment on the constructions we have given in these terms. We will distinguish between four cases, related to the four subsections earlier in this section.

**Case 1:** \( 1 < s < 2 \). We write \( s = 1 + \delta/\tau \) and choose \( d \) and \( t \) such that \( d/t = \delta/\tau \). We apply Construction \[ \text{2} \] to obtain a PIR array code with rate equal to

\[ \frac{s + 1 + \frac{1}{d}}{(2 + \frac{1}{d})s}. \]
The number of servers in this construction is \((t+d)^\vartheta \cdot \left(\frac{d}{t+1}\right)^\vartheta\), where \(\vartheta\) is the least common divisor of \(d\) and \(t\). Clearly, one should try to reduce the number of servers for these parameters.

Construction 6 yield a slightly better code. The rate of the code is the same as in Construction 2, while the number of servers used is only \(\frac{2d+1}{d+1} \cdot \left(\frac{t+d}{t}\right)^\vartheta\), which is smaller than the number of servers required by Construction 2. The disadvantage of this construction is the requirement for a Steiner system \(S(d, d+1, t+d)\) to exist. This severely limits the number of parameters that can be covered by this construction: a new construction with a small number of servers would be very desirable.

Case 2: \(s = 2\). We use Construction 1. The PIR array code will have an optimal rate \(\frac{3t+1}{4t+2}\) with \((2t+1)\) servers required for the \(k\)-PIR array protocol.

Case 3: \(s\) is an integer greater than 2. We can use Construction 9 to produce a PIR array code with rate \(\frac{(s+1)t+1}{s(2t+1)}\), which is asymptotically optimal. (Indeed, we believe the rate of this construction is optimal.) This construction requires

\[
\frac{(ts)}{t} \frac{(ts-t-1)!}{(t-s+1)!(s-2)!(t+1)!^{s-2}} + \frac{(ts)!}{(t-s+1)!(s-1)!(t+1)!^{s-1}}
\]

servers. To reduce the number of servers we can instead use Construction 7 as explained is Subsection 3.5. When \(t = s^2\) we obtain codes with rate \(\frac{s+1}{2s} - \frac{1}{s^2}\), and the number of servers used in the code is

\[
\frac{s^3!}{(s^2-s)!s!(s^2-s+1)!s^s}
\]

The rate of the code is slightly lower than in Construction 9, but we need significantly fewer servers. Similar results are obtained if we use Construction 10.

Case 4: \(s\) is rational, not an integer, and greater than 2. We can use Construction 7 or Construction 8. If we use Construction 7 then we might need to use Theorem 6 to obtain the exact value of \(s\) which is required. We demonstrate the technique with two examples in which \(4 < s < 5\).

Example 1. Assume we want to design a PIR array code for \(s = 4\frac{1}{6}\). We start with Construction 7 using \(r = \lfloor s \rfloor = 5\). If we set \(t = 24\) then \(s = 4\frac{1}{6}\) and by Theorem 7, we have \(g(4\frac{1}{6}, 24) \geq \frac{31}{52}\). Theorem 1 now implies that \(g(4\frac{1}{6}, 24) \geq \frac{31}{52}\). If we set \(t = 24\) then we obtain in a similar way \(g(4\frac{1}{6}, 24) \geq \frac{67}{112}\), as we have a code of higher rate. If we set \(t = 3 \cdot 2^i\) then we obtain \(g(4\frac{1}{6}, 3 \cdot 2^i) \geq \frac{1}{2} + \frac{3 \cdot 2^i - 4}{2(15 \cdot 2^i - 10)}\) which becomes close to \(\frac{3}{3}\) as \(i\) grows. This can be compared to the upper bound of Theorem 8 which implies \(g(4\frac{1}{6}) \leq \frac{31}{50}\). There is a gap between the bounds.
Example 2. Assume we want to design a PIR array code for $s = 4\frac{5}{6}$. We start with Construction 7 by using $r = \lceil s \rceil = 5$. If we set $t = 96$ then by using the same computation as in Example 1 we obtain $s = 4\frac{5}{6}$ and by Theorem 10 we have $g(4\frac{5}{6}, 96) \geq \frac{139}{232}$, while the upper bound of Theorem 3 implies $g(4\frac{5}{6}) \leq \frac{140}{232}$. Hence there is a gap between the upper and lower bounds on the optimal rate in this case also.

Some lessons that can be learned here: The rates of construction improve as $t$ becomes larger in Theorem 10. Also, applying Theorem 6 more often than necessary is not desirable, as this results in codes with lower rate. The analysis of Theorem 10 shows that Construction 7 is more effective as $r$ gets larger.

Construction 8 has advantages: since $s = r + d$ it is easier to characterise the pairs $(s, t)$ that the construction applied to; the rate approaches the upper bound of Theorem 3 quite fast. The disadvantage of the construction is that the number of servers used is very large. We would like to see a construction with the same rate, but a reduced number of servers.

We end this section with a discussion of the pairs $(s, t)$ which are not covered by our constructions. First, we can easily verify that if $s$ is an integer then all pairs $(s, t)$ such that $t \geq s$ are covered by our constructions. Our constructions do not consider the pairs $(s, t)$ in which $s$ is an integer and $t < s$. If $s$ is a rational number greater than one, i.e. $s = r + \frac{\delta}{\tau}$, where $1 \geq r$, $2 \leq \tau$, and $1 \leq \delta \leq \tau - 1$, then $t$ must be divisible by $\tau$ since $p = st$ (as without loss of generality we can assume that $\delta$ and $\tau$ are coprime). Therefore, $t = \tau\mu$ and hence $s = r + \frac{\delta\mu}{\tau\mu} = r + \frac{\delta\mu}{\tau}$. Constructions 2 and 8 provide solutions for pairs $(s, t)$ as long as $\tau$ divides $t$ and $t \geq r$ is at least 2. Thus, we need constructions for the cases of pairs $(s, t)$ for which $s > 2$, $\tau$ divides $t$, and $t < \lceil s \rceil$.

4 Conclusions and Problems for future research

We have constructed $k$-PIR array codes with good PIR rate for many pairs $(s, t)$, where a database is divided into $st$ parts, and each server stores $t$ linear combinations of parts in its cells. Our constructions are mainly based on set systems. We have also proved upper bounds on the PIR rate of a PIR array code. These results are strong enough to determine the best rate of a PIR array code when $1 < s \leq 2$ for all sensible choices of $t$. Moreover, the results determine the asymptotic value for the PIR rate for all rational values of $s$, when $t$ is allowed to tend to infinity. As was mentioned in [12, 13], these results have interesting implications for the size of codes used for locality and availability in distributed storage.

The research on PIR array codes is far from being complete. Some problems for future research are enumerated below. Some of these problems are currently being considered by the authors, and we hope that answers will be the subject of future work.

1. We believe that all our main constructions (Construction 1, 2, 8, and 9) yield optimal codes, i.e. codes for which the rate is the largest possible one. But when $s > 2$ our upper bounds on the optimal rate fall short of the lower bounds implied by these constructions, although the gap tends to zero as $t$ grows. We would like to see improved upper bounds on $g(s, t)$ when $s > 2$. Clearly such a new upper bound must take into account that more than one cell in some servers will have linear combinations which are not singletons. Also, it seems that the linear combinations in each cell should not
involve more than $t + 1$ parts. This should be somehow reflected in the analysis of a new upper bound.

2. Our constructions cover a large range of pairs $(s, t)$, but nevertheless there are some pairs which are not covered. These pairs are analyzed in Subsection 3.6. We would like to see new constructions as well as constructions derived from the known ones to cover the missing pairs. The optimal scenario will be to cover all the pairs $(s, t)$ with a construction which achieves the highest rate. It should be noted that known upper bounds are unlikely to be tight for all pairs $(s, t)$. In particular, the case when $t$ is small might be more practical and its consideration is a research problem for itself. The case $t = 1$, which relates to PIR codes, was considered in [12, 13], so it is of great interest to consider the next case, namely $t = 2$.

3. Constructions 6 is based on Steiner systems; it is particularly interesting as it yields optimal codes with a small number of servers. Since Steiner systems exist for only a small set of parameters, and even when they exist there is no efficient construction for most of them, we would like to have constructions with similar performance which do not depend on such systems.

4. For many pairs $(s, t)$, our constructions that achieve optimal rate have $k$ and $m$ impractically large. Hence we ask: What is the smallest $k$ for which an optimal rate can be obtained? More generally, we would like to have more detailed bounds that explain the tradeoff between the parameters $t$, $k$, and $m$, for a fixed value of $s$.

5. We have not made a detailed comparison of the values of $g(s, t)$ and $g(s', t')$ in this paper; this was done in [12] for the case when $t = 1$. Clearly such comparisons can be made using our lower and upper bounds, but much remains to be done. Questions that could be asked are as follows. If $s < s'$, when can $g(s', t')$ be larger than $g(s, t)$? Is it the case that when $t = t'$ and $s < s'$ then $g(s, t) \geq g(s', t')$? (Theorem 6 states that $g(s, t) \leq g(s - 1/t, t)$, but this is far from a full answer.) Finally, is it the case that if $s = s'$ and $t < t'$, then $g(s, t) < g(s', t')$?

6. In this paper, we regarded $s$ as fixed and asked for the best rate as $t$ varies. A different approach might be to fix $t$ and to vary values of $s$. (Indeed, this could be carried out within our framework.) Other directions include fixing parameters such as $k$.

7. We would like to develop the connections between this work and codes for distributed storage, especially to codes with locality and availability.

8. All our array codes which we constructed are binary codes, but generalizing to non-binary fields is very natural.

Acknowledgment

The authors are indebted to Eitan Yaakobi for many helpful discussions and providing his drafts during this research.
References

[1] D. Augot, F. Levy-Dit-Vahel, and A. Shikfa, A storage-efficient and robust private information retrieval scheme allowing few servers, in Cryptology and Network Security, 222–239, Springer 2014.

[2] Zs. Baranyai, On the factorization of the complete uniform hypergraph, in Infinite and Finite Sets, Proc. Coll. Keszthely, 1973, Colloquia Math. Soc. János Bolyai, 10, 91–107, North-Holland 1975.

[3] T. H. Chan, S. Ho, and H. Yamamoto Private information retrieval for coded storage, arxiv.org/abs/1410.5489, October 2014.

[4] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, Private information retrieval, Journal ACM, pp. 065–981, 1998.

[5] T. Etzion and N. Silberstein, Error-correcting codes in projective spaces via rank-metric codes and Ferrers diagrams, IEEE Trans. Inform. Theory, IT-55 (2009), 2909–2919.

[6] T. Etzion and N. Silberstein, Codes and designs related to lifted MRD codes, IEEE Trans. Inform. Theory, IT-59 (2013), 1004–1013.

[7] T. Etzion and L. Storme, Galois geometries and coding theory, Designs, Codes, and Cryptography, 78 (2016), 311–350.

[8] T. Etzion and A. Vardy, Error-correcting codes in projective space, IEEE Trans. Inform. Theory, IT-57 (2011), 1165–1173.

[9] T. Etzion and A. Wachter-Zeh, Vector network coding based on subspace codes outperforms scalar linear network coding, to be presented in ISIT2016, (2016).

[10] T. Etzion and A. Wachter-Zeh, Vector network coding based on subspace codes outperforms scalar linear network coding, arxiv.org/abs/1512.06352, February 2016.

[11] G. Fanti and K. Ramchandran, Efficient private information retrieval over unsynchronized databases, IEEE J. on Selected Topics in Signal Processing, 9 (2015), 1229–1239.

[12] A. Fazeli, A. Vardy, and E. Yaakobi, Coded for distributed PIR with low storage overhead, IEEE Int. Symp. on Inf. Theory (ISIT), 2015, 2852–2856.

[13] A. Fazeli, A. Vardy, and E. Yaakobi Private information retrieval without storage overhead: coding instead of replication, arxiv.org/abs/1505.0624, May 2015.

[14] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, On the locality of codeword symbols, IEEE Trans. Inform. Theory, IT-58 (2012), 6925–6934.

[15] S. Goparaju and R. Calderbank, A new sub-packetization bound for minimum storage regenerating codes, IEEE Int. Symp. on Inf. Theory (ISIT), 2013, 1616–1620.
[16] S. Goparaju, I. Tamo, and R. Calderbank, An improved sub-packetization bound for minimum storage regenerating codes, *IEEE Trans. Inform. Theory*, IT-60 (2014), 2770–2779.

[17] H. D. L. Hollmann, Storage codes – coding rate and repair locality, in *International Conference on Computing, Networking and Communications (ICNC)*, 2013, 830–834.

[18] H. D. L. Hollmann and W. Poh, Characterizations and construction methods for linear functional-repair storage codes, *IEEE Int. Symp. on Inf. Theory (ISIT)*, 2013, 336–340.

[19] P. Huang, E. Yaakobi, H. Uchikawa, and P. H. Siegel, Linear Locally Repairable Codes with Availability, *IEEE Int. Symp. on Inf. Theory (ISIT)*, 2015, 1871–1875.

[20] P. Keevash, The existence of designs, arxiv.org/abs/1401.3665, January 2014.

[21] R. Kötter and F. R. Kschischang, Coding for errors and erasures in random network coding, *IEEE Trans. Inform. Theory*, IT-54 (2008), 3579–3591.

[22] E. Kushilevitz and R. Ostrovsky.,, Replication is not needed: Single database, computationally-private information retrieval, *Proc. 38-th IEEE Symp. Foundations Computer Science (FOCS)*, 1997, 364–373.

[23] L. Pamies-Juarez, H. D. L. Hollmann, and F. Oggier, Locally repairable codes with multiple repair alternatives, *IEEE Int. Symp. on Inf. Theory (ISIT)*, 2013, 892–896.

[24] N. Raviv, T. Etzion, Distributed storage systems based on intersecting subspace codes, *IEEE Int. Symp. Inf. Theory (ISIT)*, 2015, 1462–1466.

[25] N. Raviv, N. Silberstein, and T. Etzion, Constructions of high-rate minimum storage regenerating codes over small fields, to be presented in ISIT2016, (2016).

[26] A. Rawat, D. Papailiopoulos, A. Dimakis, and S. Vishwanath, Locality and availability in distributed storage, *IEEE Int. Symp. Inf. Theory (ISIT)*, 2014, 681–685.

[27] N. Shah, K. Rashmi, and K. Ramchandran,, One extra bit of download ensures perfectly private information retrieval, *IEEE Int. Symp. Inf. Theory (ISIT)*, 2014, 856–860.

[28] R. Tajeddine and Salim El Rouayeb, Private information retrieval from MDS coded data in distributed storage systems, to be presented in ISIT2016, (2016).

[29] R. Tajeddine and Salim El Rouayeb, Private information retrieval from MDS coded data in distributed storage systems, arxiv.org/abs/1602.01458, February 2016.

[30] I. Tamo and A. Barg, Bounds on locally recoverable codes with multiple recovering sets, *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, 2014, 691–695.
[31] I. Tamo and A. Barg, A family of optimal locally recoverable codes, *IEEE Trans. Inform. Theory*, IT-60 (2014), 4661–4676.

[32] I. Tamo, Z. Wang, and J. Bruck, Long MDS codes for optimal repair bandwidth, *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, 2012, 1182-1186.

[33] I. Tamo, Z. Wang, and J. Bruck, Access versus bandwidth in codes for storage, *IEEE Trans. Inform. Theory*, IT-60 (2014), 2026–2037.

[34] A. Wang, Z. Zhang, and M. Liu, Achieving arbitrary locality and availability in binary codes, *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, 2015, 1866–1870.