Private Information Retrieval Without Storage Overhead: Coding Instead of Replication

Alexander Vardy and Eitan Yaakobi, *Senior Member, IEEE*

***Abstract*—Private information retrieval (PIR) protocols allow a user to retrieve a data item from a database without revealing any information about the identity of the item being retrieved. Specifically, in information-theoretic k-server PIR, the database is replicated among k non-communicating servers, and each server learns nothing about the item retrieved by the user. The effectiveness of PIR protocols is usually measured in terms of their *communication complexity*, which is the total number of bits exchanged between the user and the servers. However, another important cost parameter is *storage overhead*, which is the ratio between the total number of bits stored on all the servers and the number of bits in the database. Since single-server information-theoretic PIR is impossible, the storage overhead of all existing PIR protocols is at least 2 (or k, in the case of k-server PIR). In this work, we show that information-theoretic PIR can be achieved with storage overhead arbitrarily close to the optimal value of 1, without sacrificing the communication complexity asymptotically. Specifically, we prove that all known linear k-server PIR protocols can be efficiently emulated, while preserving both privacy and communication complexity but significantly reducing the storage overhead. To this end, we distribute the n bits of the database among $s+r$ servers, each storing n/s *coded bits* (rather than replicas). Notably, our coding scheme remains the same, regardless of the specific k-server PIR protocol being emulated. For every fixed k, the resulting storage overhead $(s+r)/s$ approaches 1 as s grows; explicitly we have $r \le k\sqrt{s}(1+o(1))$. Moreover, in the special case $k=2$, the storage overhead is only $1+\frac{1}{s}$. In order to achieve these results, we introduce and study a new kind of binary linear codes, called here *k-server PIR codes*. We then show how such codes can be constructed from one-step majority-logic decodable codes, from Steiner systems, from constant-weight codes, and from certain locally recoverable codes. We also establish several bounds on the parameters of k-server PIR codes and finally extend for array codes.**

***Index Terms*—Private information retrieval, availability codes, codes with locality, privacy.**

Manuscript received 6 February 2023; revised 19 April 2023; accepted 8 June 2023. Date of publication 12 July 2023; date of current version 29 August 2023. This work was supported in part by the Israel Science Foundation under Grant 1817/18; in part by the Technion Hiroshi Fujiwara Cyber Security Research Center; and in part by the Israel National Cyber Directorate. This article was presented in part at the IEEE International Symposium on Information Theory (ISIT), Hong Kong, June 2015 [DOI: 10.1109/ISIT.2015.7282977]. *(Corresponding author: Eitan Yaakobi.)*

Alexander Vardy, deceased, was with the Department of Electrical and Computer Engineering, the Department of Computer Science and Engineering, and the Department of Mathematics, University of California at San Diego, La Jolla, CA 92093 USA.

Eitan Yaakobi is with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: yaakobi@cs.technion.ac.il).

Digital Object Identifier 10.1109/JSAIT.2023.3285665

I. Introduction

PRIVATE information retrieval protocols make it possible to retrieve a data item from a database without disclosing any information about the identity of the item being retrieved. This may be thought of as a weaker version of 1-out-of-n oblivious transfer, where it is also required that the user learns nothing about the other database items. The notion of private information retrieval (PIR) was first introduced by Chor et al. in [11], [12] and has attracted considerable attention since (see [8], [9], [14], [18], [46], [47] and references therein). The classic PIR model of [12], which we adopt in this paper, views the database as a binary string $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ and assumes that the user wishes to retrieve a single bit x_i without revealing any information about the index i. A naive solution for the user is to download the entire database $\boldsymbol{x}$. It is shown in [12] that in the case of a single database stored on a single computationally-unbounded server, this solution is essentially the best possible: any PIR protocol will require $\Omega(n)$ bits of communication between the user and the server. In order to achieve sublinear communication complexity, Chor et al. [12] proposed *replicating the database* on several servers that do not communicate with each other. They showed that having two replicas makes it possible to reduce the communication cost to $O(n^{1/3})$, while $k \ge 3$ servers can achieve $O((k^2 \log k) n^{1/k})$ communication complexity.

Following the seminal work of [12], the communication complexity of information-theoretic k-server PIR has been further reduced in a series of groundbreaking papers. Ambainis [2] generalized the methods of [12] to obtain a communication cost of $O(n^{1/(2k-1)})$ for all $k \ge 2$. This result remained the best known until the $O(n^{1/(2k-1)})$-complexity barrier was finally broken in [8]. Five years later, came the remarkable work of Yekhanin [47] who constructed a 3-server PIR scheme with subpolynomial communication cost, assuming the infinitude of Mersenne primes. Shortly thereafter, Efremenko [15] gave an unconditional k-server PIR scheme with subpolynomial complexity for all $k \ge 3$. The recent paper of Dvir and Gopi [14] shows how to achieve the same complexity as in [15] with only two servers.

All this work follows the original idea, first proposed in [12], of replicating the database in order to reduce the communication cost. However, this approach neglects another cost parameter: the *storage overhead*, defined as the ratio between the total number of bits stored on all the servers and the number of bits in the database. Clearly, the storage overhead of all the PIR protocols discussed above is $k \ge 2$. If the database

is very large, the necessity to store several replicas of it could be untenable for some applications. Thus, in this paper, we consider the following question. Can one achieve PIR with low communication cost but *without doubling (or worse) the number of bits we need to store*?

This question has been settled in the affirmative in [25] for the case where one is willing to replace information-theoretic guarantees of privacy by computational guarantees. Such computational PIR is by now well studied — see [18], [25] for more information. However, in this paper, we consider only information-theoretic PIR, which provides the strongest form of privacy. That is, even computationally unbounded servers should not gain any information from the user queries. Somewhat surprisingly, despite the impossibility proof of [12], the answer to our question turns out to be affirmative also in the case of information-theoretic PIR. We note that our results do not contradict [12]. To achieve information-theoretic privacy, one does need $k \geq 2$ non-communicating servers. However, these servers do not have to hold the entire database — they can store only parts of it. We show that if these parts are *judiciously encoded, rather than replicated*, the overall storage overhead can be reduced. Asymptotically, as the number of such parts grows, storage overhead can be eliminated altogether.

A. Our Contributions

We show that all known k-server information-theoretic PIR protocols can be efficiently emulated, while preserving their privacy and communication-complexity guarantees (up to a small constant), but significantly reducing the storage overhead. In fact, for any fixed k and any $\epsilon > 0$, we can reduce the storage overhead to under $1 + \epsilon$.

In order to achieve these results, we first partition the database into s parts and distribute these parts among non-communicating servers, so that every server stores n/s bits. Why do we partition the database in this manner? The main reason is that such partition is necessary to reduce the storage overhead. If every server has to store all n bits of the database, then the storage overhead cannot be reduced beyond $k \geq 2$. However, in practice, there may be other compelling reasons. For example, the database may be simply too large to fit in a single server, or it may need to be stored in a distributed manner for security purposes.

We observe that the number of parts s need not be very large. With $s = 2$ parts, we can already achieve significant savings in storage overhead. With $s = 16$ parts and $m = 17$ servers, we get a storage overhead of 1.0625.

Given a partition of the database into s parts, our construction uses two main ingredients: 1) an existing k-server PIR protocol in which the servers' responses are a *linear function* of the database bits, and 2) a binary linear code, which we call a *k-server PIR code*, with a special property to be defined shortly. We note that the first requirement is very easy to satisfy. All the existing PIR protocols known (to us) are linear in this fashion. Therefore our primary focus in this paper is on the construction of k-server PIR codes.

The defining property of a k-server PIR code is this: for every message bit x_i, there exist k disjoint sets of coded bits from which x_i can be uniquely recovered (see Section III for a formal definition). Although this property is reminiscent of locally recoverable codes, introduced in [19], there are important differences. In locally recoverable codes, we wish to guarantee that every message bit x_i can be recovered from a *small set* of coded bits, and only one such recovery set is needed. Here, we wish to have k *disjoint recovery sets* for every message bit, and we do not care about their size. To the best of our knowledge, codes with this property have not been previously studied. Such codes may be of independent interest, especially in distributed storage applications.

In this paper, we show how k-server PIR codes can be constructed from Steiner systems, from one-step majority-logic decodable codes, from bipartite graphs of girth 6, and from constant-weight codes. We give an optimal construction of such codes when *either* k or the number of parts s is small. We also establish several upper and lower bounds on the parameters of general k-server PIR codes.

B. Related Work

Coding to reduce storage overhead while emulating conventional PIR protocols is a new idea, first proposed in this paper. Nevertheless, we are aware of several earlier papers [3], [9], [33], [35] that construct and study certain kinds of coded PIR schemes. Shah et al. [33] show how to encode and distribute files among multiple servers in order to guarantee private retrieval with very low communication complexity. However, the setting considered in [33] is quite different from the standard PIR model of [12]. Moreover, the methods of [33] require an exponentially large number of servers, which may depend on the number of files stored and/or their size. Chan et al. [9] study the tradeoff between storage overhead and communication complexity, but only for the case where the size of each data item stored is very large (rather than one bit, as in [12] and this paper). Sun and Jafar [35] follow-up on the results of [9] and determine the *PIR capacity*, defined in terms of the number of private information bits retrieved per each downloaded bit. The coding schemes proposed in [9], [35] do *not* necessarily reduce storage overhead. Moreover, the regime considered in these papers differs significantly from the accepted PIR model of [11], [12]. The work of Augot et al. [3] is contemporaneous with ours and is the closest to our results herein. The authors of [3] consider specifically the PIR protocol based on the multiplicity codes of Kopparty et al. [23]. They use the geometry of multiplicity codes in order to avoid full replication of the database on each server. The resulting storage overhead is $1/R$, where R is the rate of the underlying multiplicity code. This is usually much higher than the storage overheads achieved in this paper for a comparable communication complexity. Most importantly, the results of [3] are limited to a specific PIR protocol based on multiplicity codes, whereas our results are generic: we can emulate *any* existing PIR protocol while reducing its storage overhead.

Another notion closely related to our work is that of *multiset batch codes*. Batch codes were introduced in [21] in order to efficiently balance the read-out load in distributed storage systems. Specifically, a batch code encodes a binary string $\boldsymbol{x}$

into an m-tuple of strings, one for each of m servers, so that any multiset (a.k.a. batch) of k bits from $\boldsymbol{x}$ can be decoded by reading at most one bit from each server. Following [21], batch codes have been extensively studied; for example, see [31] and references therein. In particular, they found applications in PIR and related cryptographic protocols. Herein, we observe that a multiset batch code is also a k-server PIR code, but not vice versa. For more details on the precise relationship between our PIR codes and batch codes, see [41].

C. Organization

The rest of this paper is organized as follows. In the next section, we formally define k-server PIR protocols and introduce the linearity property needed for our constructions. We then consider a specific toy example of 3-server PIR, which serves to motivate our discussion in the following section. Generalizing from this example, we propose the formal notion of *coded PIR protocols*. In Section III, we prove that any conventional PIR protocol can be emulated by a coded PIR protocol with reduced storage overhead, provided it is linear. The proof hinges on the existence of certain binary linear codes, called *k-server PIR codes*, that are formally defined in Section III. Section IV is devoted to explicit constructions of k-server PIR codes. We show that such codes can be obtained from a variety of well-studied objects in combinatorics and coding theory. These include Steiner systems, majority-logic decodable codes, bipartite graphs of girth 6, and constant-weight codes. In each case, we analyze the redundancy of the resulting codes, which determines the storage overhead of the corresponding coded PIR schemes. In Section V, we provide upper and lower bounds on this redundancy for the case where both k and the code dimension s are small. We furthermore determine how the redundancy of k-server PIR codes behaves asymptotically, when either s or k is fixed while the other parameter tends to infinity. Section VI studies the asymptotic behavior of coded PIR and Section VII studies PIR codes as array codes.

II. Definitions and Preliminaries

We begin with a formal definition of a PIR protocol, satisfying information-theoretic privacy requirements. We generally follow the conventional notions of PIR, as introduced in [11], [12] and further elaborated upon in [18] and [46].

Definition 1: A *k-server PIR scheme* consists of the following: a binary string $\boldsymbol{x}$ of length n, called the *database*, k non-communicating *servers* $\mathcal{S}_1, \ldots, \mathcal{S}_k$ each storing a copy of $\boldsymbol{x}$, a *user* (hereinafter, called Alice) who wishes to retrieve x_i for some $i \in [n]$, without revealing i to any of the servers, and a *k-server PIR protocol*. The k-server PIR protocol $\mathcal{P}$ involves a triple of algorithms $\langle \mathcal{Q}, \mathcal{A}, \mathcal{C} \rangle$ and consists of the following steps:

- *Step 1:* Alice flips random coins and uses the outcome to invoke the *query algorithm* $\mathcal{Q}(k, n; i)$ that generates a k-tuple of queries $(q_1, \ldots, q_k)$. For $j \in [k]$, the query q_j will be also denoted by $\mathcal{Q}_j(k, n; i)$.
- *Step 2:* For all $j \in [k]$, Alice sends the query q_j to the j-th server $\mathcal{S}_j$.
- *Step 3:* For all $j \in [k]$, the server $\mathcal{S}_j$ invokes the *answer algorithm* $\mathcal{A}$ to respond with the answer $a_j = \mathcal{A}(k, j; \boldsymbol{x}, q_j)$.
- *Step 4:* Alice computes her output by invoking the *reconstruction algorithm* $\mathcal{C}(k, n; i, a_1, \ldots, a_k)$.

The answer algorithm $\mathcal{A}$ and the reconstruction algorithm $\mathcal{C}$ are deterministic, while the query algorithm $\mathcal{Q}$ is randomized (the random input to $\mathcal{Q}$ is suppressed for notational convenience). The k-server PIR protocol $\mathcal{P}$ must satisfy the following correctness and privacy requirements:

Correctness: For all $\boldsymbol{x} \in \{0,1\}^n$ and $i \in [n]$, Alice correctly determines x_i. That is, $\mathcal{C}(k, n; i, a_1, \ldots, a_k) = x_i$.

Privacy: Each server $\mathcal{S}_j$ learns no information about i. That is, for all $i_1, i_2 \in [n]$, the distributions of $\mathcal{Q}_j(k, n; i_1)$ and $\mathcal{Q}_j(k, n; i_2)$ are identical, where the distribution is over the coins flips in Step 1. This must hold for all $j \in [k]$.

The queries $q_1, \ldots, q_k$ and the answers $a_1, \ldots, a_k$ are all assumed to be binary strings. The *communication complexity* of a PIR protocol is defined as the (worst-case) sum of their lengths $|q_1| + \cdots + |q_k| + |a_1| + \cdots + |a_k|$. Following [9], [21], [33] and other papers, the *storage overhead* of a PIR scheme is defined as the ratio between the total number of bits stored on all the servers and the number of bits in the database. It is obvious that the storage overhead of any k-server PIR scheme, as defined above, is k.

Following the foundational work of Chor et al. [11], [12], numerous variations on Definition 1 have been studied over the years. These include computational PIR (with computational rather than information theoretic privacy guarantees), symmetric PIR (with privacy guarantees for the servers as well as the user), private information storage (with privacy guarantees for both read and write operations), robust PIR (tolerant against non-responsive servers), Byzantine PIR (tolerant against malicious servers), t-private PIR (tolerant against collusions of up to t servers), PIR in the random server model, practical PIR, quantum PIR, and others. Although our results herein extend to many of these scenarios, the discussion of such extensions is beyond the scope of the present paper.

On the other hand, our constructions require PIR protocols satisfying a certain additional property that, to the best of our knowledge, has not been so far considered in the literature. This property is *database-linearity*, or simply linearity, formally defined as follows.

Definition 2: We say that a k-server PIR protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ is *linear* if for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \{0,1\}^n$ and for all possible queries $q_1, \ldots, q_k$, the following holds:

$$\mathcal{A}(k, j; \boldsymbol{x}_1 + \boldsymbol{x}_2, q_j) = \mathcal{A}(k, j; \boldsymbol{x}_1, q_j) + \mathcal{A}(k, j; \boldsymbol{x}_2, q_j) \text{ for all } j \in [k] \quad (1)$$

Note that $\mathcal{A}$ is the only algorithm in the triple $\langle \mathcal{Q}, \mathcal{A}, \mathcal{C} \rangle$ that explicitly depends on the database $\boldsymbol{x}$. Thus the linearity of $\mathcal{A}$ makes the whole protocol linear. Fortunately, *many the known (to us) PIR protocols* that are targeted to minimize the upload and download complexity satisfy the linearity condition of Definition 2. In particular, this is true for all the protocols considered in [7], [8], [12], [14], [46], [47].

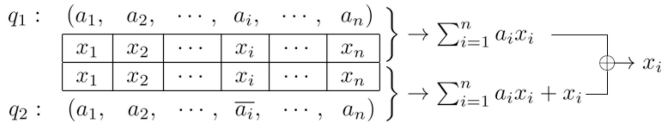

Fig. 1. Alice sends $q_1 = \boldsymbol{a}$ and $q_2 = \boldsymbol{a} + \boldsymbol{e}_i$ to the servers. The servers respond with $\boldsymbol{a} \cdot \boldsymbol{x}$ and $(\boldsymbol{a} + \boldsymbol{e}_i) \cdot \boldsymbol{x}$ and Alice recovers x_i as their sum. The value of i remains private as the vector $\boldsymbol{a}$ is chosen uniformly at random.

Lastly, we wish to make explicit an assumption tacitly made in Definition 1, since it will be needed in what follows. It is assumed that the answer algorithm $\mathcal{A}$ is public knowledge. This, in particular, implies that given a query q, every server can compute the response $\mathcal{A}(k, j; \boldsymbol{x}, q)$ for all $j \in [k]$. That is, any server can simulate any other server.

We are now ready to proceed with two examples of coded version of PIR protocols, designed to reduce the storage overhead. The first example is for a 2-server protocol while the second, which best illustrates some of the main ideas in this paper, is designed for a 3-server protocol.

Example 1: Consider the following 2-server PIR scheme where each server stores an n-bit database $\boldsymbol{x}$ and Alice wants to read the i-th bit x_i, for some $i \in [n]$. Alice chooses uniformly at random a vector $\boldsymbol{a} \in \{0,1\}^n$. The first server receives the query $\boldsymbol{a}$ and responds with an answer of the bit $\boldsymbol{a} \cdot \boldsymbol{x}$. The second server receives the query $(\boldsymbol{a} + \boldsymbol{e}_i)$ and responds with an answer of the bit $(\boldsymbol{a} + \boldsymbol{e}_i) \cdot \boldsymbol{x}$; see Fig. 1. Alice receives these two bits and their sum gives the i-th bit x_i, since

$$\boldsymbol{a} \cdot \boldsymbol{x} + (\boldsymbol{a} + \boldsymbol{e}_i) \cdot \boldsymbol{x} = \boldsymbol{a} \cdot \boldsymbol{x} + \boldsymbol{a} \cdot \boldsymbol{x} + \boldsymbol{e}_i \cdot \boldsymbol{x} = x_i.$$

If the servers do not communicate with each other then since the vector $\boldsymbol{a}$ is chosen uniformly at random, the value of i remains private. Moreover, the servers' responses are linear functions of the stored data and thus the protocol is a linear PIR protocol. Alice had to transmit $2n$ bits and 2 bits were received, so a total of $2n + 2$ bits were communicated. The storage overhead of this scheme is 2 and note also that if one of the servers fails then it is possible to retrieve the database $\boldsymbol{x}$ from the other surviving server.

Now, assume that the database $\boldsymbol{x}$ is partitioned into two equal parts of $n/2$ bits each, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, where $\boldsymbol{x}_1 = (x_1, \ldots, x_{n/2})$, and $\boldsymbol{x}_2 = (x_{n/2+1}, \ldots, x_n)$. The database is stored in three servers. The first server stores $\boldsymbol{x}_1$, the second stores $\boldsymbol{x}_2$, and the third one is a parity server which stores $\boldsymbol{x}_1 + \boldsymbol{x}_2$. If Alice wants to read the i-th bit where $i \in [n/2]$, she first chooses uniformly at random a vector $\boldsymbol{a} \in \{0,1\}^{n/2}$. The first server receives the query $\boldsymbol{a}$ and responds with the bit $\boldsymbol{a} \cdot \boldsymbol{x}_1$. The second server receives the query $\boldsymbol{a} + \boldsymbol{e}_i$ and responds with the bit $(\boldsymbol{a} + \boldsymbol{e}_i) \cdot \boldsymbol{x}_2$, and the third server receives the query $\boldsymbol{a} + \boldsymbol{e}_i$ and responds with the bit $(\boldsymbol{a} + \boldsymbol{e}_i) \cdot (\boldsymbol{x}_1 + \boldsymbol{x}_2)$. Alice receives those three bits and calculates the bit x_i according to the sum

$$\begin{aligned} &\boldsymbol{a} \cdot \boldsymbol{x}_1 + (\boldsymbol{a} + \boldsymbol{e}_i) \cdot \boldsymbol{x}_2 + (\boldsymbol{a} + \boldsymbol{e}_i) \cdot (\boldsymbol{x}_1 + \boldsymbol{x}_2) \\ &\quad = \boldsymbol{a} \cdot \boldsymbol{x}_1 + (\boldsymbol{a} + \boldsymbol{e}_i) \cdot \boldsymbol{x}_1 = x_i. \end{aligned}$$

It is clear that both schemes keep the privacy of i. In the first scheme, the number of communicated bits is $2n+2$, while in the coded scheme it is $3n/2 + 3$. The storage overhead was improved from 2 to 3/2, and both schemes can tolerate a single server failure. However, we note that the coded scheme requires one more server.

Example 2: Let $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ denote your favorite 3-server PIR protocol, and assume that it is linear. Suppose you are given a database string $\boldsymbol{x}$ of length n that is too large to fit in a single server. In fact, let us assume that each server can store at most $n/4$ of the n bits. What shall you do? Of course, you can partition $\boldsymbol{x}$ into four equal parts $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$, store 3 replicas of each part, and then apply the protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ essentially unchanged. Altogether, you will need 12 servers, each storing $n/4$ bits, and the resulting storage overhead is 3, as expected. The point of this example is that you can do much better: you can achieve a storage overhead of 2 rather than 3, using only 8 servers.

As before, we begin with a partition of the database $\boldsymbol{x}$ into four equal parts $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$. However, instead of replicating them, we shall *encode* these four parts as follows:

$$\begin{aligned} &\boldsymbol{c}_1 = \boldsymbol{x}_1,\ \boldsymbol{c}_2 = \boldsymbol{x}_2,\ \boldsymbol{c}_3 = \boldsymbol{x}_3,\ \boldsymbol{c}_4 = \boldsymbol{x}_4,\ \boldsymbol{c}_5 = \boldsymbol{x}_1 + \boldsymbol{x}_2, \\ &\boldsymbol{c}_6 = \boldsymbol{x}_2 + \boldsymbol{x}_3,\ \boldsymbol{c}_7 = \boldsymbol{x}_3 + \boldsymbol{x}_4,\ \boldsymbol{c}_8 = \boldsymbol{x}_4 + \boldsymbol{x}_1 \end{aligned} \tag{2}$$

The coded shares $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_8$ are distributed among 8 non-communicating servers $\mathcal{S}_1, \ldots, \mathcal{S}_8$ (in this order). How does information retrieval work in this situation? Assume, for the time being, that Alice wishes to read the i-th bit from the first part $\boldsymbol{x}_1$. That is, she is interested in the bit $x_{1,i}$, where $i \in [n/4]$. She will first invoke the query algorithm $\mathcal{Q}$ to generate three randomized queries $(q_1, q_2, q_3) = \mathcal{Q}(3, n/4; i)$. She will then send queries to the servers as follows:

$$\begin{aligned} &(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \\ &\quad \leftarrow\ (q_1, q_2, q_3, q_3, q_2, q_2, q_3, q_3) \end{aligned}$$

The privacy property of $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ guarantees that the servers learn nothing about i from the queries (in fact, for most existing PIR protocols, q_1, q_2, q_3 are uniformly random over all binary strings of the same length). Upon receiving the servers' responses, Alice ignores the answers from $\mathcal{S}_3, \mathcal{S}_6, \mathcal{S}_7$, but collects the other five answers given by:

Server	Query	Response
$\mathcal{S}_1$	q_1	$a_1 = \mathcal{A}(3, 1; \boldsymbol{x}_1, q_1)$
$\mathcal{S}_2$	q_2	$a_2 = \mathcal{A}(3, 2; \boldsymbol{x}_2, q_2)$
$\mathcal{S}_4$	q_3	$a_4 = \mathcal{A}(3, 3; \boldsymbol{x}_4, q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3, 2; \boldsymbol{x}_1 + \boldsymbol{x}_2, q_2)$
$\mathcal{S}_8$	q_3	$a_8 = \mathcal{A}(3, 3; \boldsymbol{x}_4 + \boldsymbol{x}_1, q_3)$

(3)

Since the PIR protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ is linear, Alice can now compute:

$$\begin{aligned} a'_2 &= a_2 + a_5 = \mathcal{A}(3, 2; \boldsymbol{x}_2, q_2) + \mathcal{A}(3, 2; \boldsymbol{x}_1 + \boldsymbol{x}_2, q_2) \\ &= \mathcal{A}(3, 2; \boldsymbol{x}_1, q_2) \\ a'_3 &= a_4 + a_8 = \mathcal{A}(3, 3; \boldsymbol{x}_4, q_3) + \mathcal{A}(3, 3; \boldsymbol{x}_4 + \boldsymbol{x}_1, q_3) \\ &= \mathcal{A}(3, 3; \boldsymbol{x}_1, q_3). \end{aligned}$$

Finally, Alice sets $a'_1 = a_1$ and invokes the original reconstruction algorithm $\mathcal{C}$ with a'_1, a'_2, a'_3 to retrieve the value of

$x_{1,i}$ as follows:

$$\begin{aligned}&\mathcal{C}\big(3, n/4; i, a'_1, a'_2, a'_3\big)\\ &= \mathcal{C}\big(3, n/4; i, \mathcal{A}(3, 1; \boldsymbol{x}_1, q_1), \mathcal{A}(3, 2; \boldsymbol{x}_1, q_2), \mathcal{A}(3, 3; \boldsymbol{x}_1, q_3)\big)\\ &= x_{1,i}.\end{aligned}$$

Now assume that Alice wishes to read the i-th bit from the second part $\boldsymbol{x}_2$, namely the bit $x_{2,i}$ for some $i \in [n/4]$. She generates the queries $(q_1, q_2, q_3) = \mathcal{Q}(3, n/4; i)$ exactly as before, and sends them to the 8 servers as follows:

$$\begin{aligned}&(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8)\\ &\quad\leftarrow\ (q_2, q_1, q_3, q_3, q_2, q_3, q_3, q_3)\end{aligned}$$

She then collects the answers:

Server	Query	Response
$\mathcal{S}_1$	q_2	$a_1 = \mathcal{A}(3, 2; \boldsymbol{x}_1, q_2)$
$\mathcal{S}_2$	q_1	$a_2 = \mathcal{A}(3, 1; \boldsymbol{x}_2, q_1)$
$\mathcal{S}_3$	q_3	$a_3 = \mathcal{A}(3, 3; \boldsymbol{x}_3, q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3, 2; \boldsymbol{x}_1 + \boldsymbol{x}_2, q_2)$
$\mathcal{S}_6$	q_3	$a_6 = \mathcal{A}(3, 3; \boldsymbol{x}_2 + \boldsymbol{x}_3, q_3)$

(4)

and computes:

$$\begin{aligned}a'_2 &= a_1 + a_5 = \mathcal{A}(3, 2; \boldsymbol{x}_1, q_2) + \mathcal{A}(3, 2; \boldsymbol{x}_1 + \boldsymbol{x}_2, q_2)\\ &= \mathcal{A}(3, 2; \boldsymbol{x}_2, q_2)\\ a'_3 &= a_3 + a_6 = \mathcal{A}(3, 3; \boldsymbol{x}_3, q_3) + \mathcal{A}(3, 3; \boldsymbol{x}_2 + \boldsymbol{x}_3, q_3)\\ &= \mathcal{A}(3, 3; \boldsymbol{x}_2, q_3)\end{aligned}$$

With $a'_1 = a_2$, we now have $\mathcal{C}(3, n/4; i, a'_1, a'_2, a'_3) = x_{2,i}$. It is not difficult to verify that retrieving the i-th bit of $\boldsymbol{x}_3$ or $\boldsymbol{x}_4$ works out in a similar fashion. We leave the details of this as an exercise for the reader.

We point out, however, the following problem with (3) and (4). Consider, for example, the server $\mathcal{S}_1$. We see that upon receiving a query q, this server needs to return $\mathcal{A}(3, 1; \boldsymbol{x}_1, q)$ in (3) and $\mathcal{A}(3, 2; \boldsymbol{x}_1, q)$ in (4). As explained earlier, the server can compute both values. But how does it know which one of the two it should send to Alice? For many existing PIR protocols, the output of the answer algorithm $\mathcal{A}(k, j; \boldsymbol{x}, q)$ in fact does not depend on j, in which case the problem disappears. However, we do *not* assume that this is always the case. Note that Alice cannot simply send the appropriate index j to each server, since this may reveal information about which part of the database (e.g., $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$) she is interested in. A potential solution is to require each server to return *all k possible values*, namely

$$\mathcal{A}(k, 1; \boldsymbol{x}, q),\ \mathcal{A}(k, 2; \boldsymbol{x}, q),\ \ldots,\ \mathcal{A}(k, k; \boldsymbol{x}, q) \tag{5}$$

This works, but increases the communication complexity of the download by a factor of k. A better solution, which does not increase the download complexity, will be presented in Theorem 2 of the next section. ■

The basic premise in Example 2 is this: a database string of length n is partitioned into 4 parts, each of length $n/4$; these parts are then encoded into 8 shares distributed among non-communicating servers. With appropriate encoding, this makes it possible to reduce the storage overhead of a 3-server PIR protocol from 3 to 8/4. Guided by this example, and following closely Definition 1, let us now formally define what we mean by a coded PIR scheme.

Definition 3: A *coded PIR scheme* with s *parts* and m *shares* consists of the following: a binary string $\boldsymbol{x}$ of length n, called the database, that is partitioned into s parts $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s$, each of length n/s, m shares $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m$ of length n/s, where $\boldsymbol{c}_j$ is a linear function of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s$ for all $j \in [m]$, stored in m non-communicating servers $\mathcal{S}_1, \ldots, \mathcal{S}_m$, a user Alice who wishes to retrieve the bit x_i for some $i \in [n]$, without revealing i to any of the servers, and a *coded PIR protocol*. The coded PIR protocol $\mathcal{P}^*(\mathcal{Q}^*, \mathcal{A}^*, \mathcal{C}^*)$ consists of the following steps:

- *Step 1:* Alice flips random coins and uses the outcome to invoke the query algorithm $\mathcal{Q}^*(m, s, n; i)$ that generates an m-tuple of queries $(q_1^*, \ldots, q_m^*)$. For $j \in [m]$, the query q_j^* will be also denoted by $\mathcal{Q}_j^*(m, s, n; i)$.
- *Step 2:* For all $j \in [m]$, Alice sends the query q_j^* to the j-th server $\mathcal{S}_j$.
- *Step 3:* For all $j \in [m]$, the server $\mathcal{S}_j$ invokes the answer algorithm $\mathcal{A}^*$ to compute the answer $a_j = \mathcal{A}^*(m, s, j; \boldsymbol{c}_j, q_j^*)$.
- *Step 4:* Alice computes her output by invoking the reconstruction algorithm $\mathcal{C}^*(m, s, n; i, a_1, \ldots, a_m)$.

The coded PIR protocol $\mathcal{P}^*$ must satisfy the correctness and privacy conditions of Definition 1. That is, we require that $\mathcal{C}^*(m, s, n; i, a_1, \ldots, a_m) = x_i$ and for all i_1, i_2, the distributions of $\mathcal{Q}_j^*(m, s, n; i_1)$ and $\mathcal{Q}_j^*(m, s, n; i_2)$ are identical.

In a coded PIR scheme with s parts and m shares, we use m servers, each storing n/s bits, to encode a database of length n. Thus the corresponding storage overhead is $mn/sn = m/s$. In the next two sections, we will show that given any linear k-server PIR protocol $\mathcal{P}$, it is always possible to construct a coded PIR scheme that emulates $\mathcal{P}$ while its storage overhead m/s is arbitrarily close to 1.

III. Construction of Coded PIR Schemes

In this section, we give a general method to construct coded PIR schemes. The main idea of our construction is to use existing (linear) PIR protocols and emulate them in a coded setup. The resulting coded PIR schemes thus inherit the privacy guarantees, the correctness, and the communication complexity (up to a small factor, discussed later in this section) of the emulated PIR protocol, while significantly reducing its storage overhead.

Let us begin by revisiting Example 2. What properties of the encoding equations (2) make the resulting coded PIR scheme work? In order to answer this question, let us first rewrite (2) in a matrix form:

$$\begin{aligned}&(\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \boldsymbol{c}_5, \boldsymbol{c}_6, \boldsymbol{c}_7, \boldsymbol{c}_8)\\ &\quad= (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4)\begin{bmatrix}1&0&0&0&1&0&0&1\\0&1&0&0&1&1&0&0\\0&0&1&0&0&1&1&0\\0&0&0&1&0&0&1&1\end{bmatrix}.\end{aligned} \tag{6}$$

With reference to the matrix above, it is easy to see that each part $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$ of the database can be recovered from the coded shares $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \boldsymbol{c}_5, \boldsymbol{c}_6, \boldsymbol{c}_7, \boldsymbol{c}_8$ in $k = 3$ different ways. Explicitly, we have:

$$\boldsymbol{x}_1 = \boldsymbol{c}_1 = \boldsymbol{c}_5 + \boldsymbol{c}_2 = \boldsymbol{c}_8 + \boldsymbol{c}_4 \tag{7}$$

$$\boldsymbol{x}_2 = \boldsymbol{c}_2 = \boldsymbol{c}_5 + \boldsymbol{c}_1 = \boldsymbol{c}_6 + \boldsymbol{c}_3 \tag{8}$$

$$\boldsymbol{x}_3 = \boldsymbol{c}_3 \;=\; \boldsymbol{c}_6 + \boldsymbol{c}_3 \;=\; \boldsymbol{c}_7 + \boldsymbol{c}_4 \tag{9}$$

$$\boldsymbol{x}_4 = \boldsymbol{c}_4 \;=\; \boldsymbol{c}_7 + \boldsymbol{c}_3 \;=\; \boldsymbol{c}_8 + \boldsymbol{c}_1 \tag{10}$$

Moreover, each share $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \boldsymbol{c}_5, \boldsymbol{c}_6, \boldsymbol{c}_7, \boldsymbol{c}_8$ appears in each of the recovery equations (7)–(10) no more than once. It turns out that this is the key property that we need. It is formalized in the definitions that follow.

Definition 4: Let $\boldsymbol{e}_i$ denote the binary (column) vector with 1 in position i and zeros elsewhere. We say that an $s \times m$ binary matrix G has *property* A_k if for all $i \in [s]$, there exist k disjoint sets of columns of G that add up to $\boldsymbol{e}_i$. A matrix that has property A_k is also said to be a *k-server PIR matrix*.

Definition 5: A binary linear code $\mathbb{C}$ of length m and dimension s will be called a *k-server PIR code* if there exists an $s \times m$ generator matrix G for $\mathbb{C}$ with property A_k.

Lemma 1: Let $\mathbb{C}$ be a k-server PIR code of length m and dimension s, and suppose G is a generator matrix for $\mathbb{C}$ with property A_k. Let $\boldsymbol{c} = \boldsymbol{x}G$ be the encoding of a message $\boldsymbol{x} = (x_1, \ldots, x_s) \in \{0,1\}^s$. Then for all $i \in [s]$, there exist k disjoint *recovery sets* $\mathcal{R}_1, \ldots, \mathcal{R}_k \subseteq [m]$ such that

$$x_i \;\coloneqq\; \sum\nolimits_{j \in \mathcal{R}_1} c_j \;=\; \cdots \;=\; \sum\nolimits_{j \in \mathcal{R}_k} c_j \tag{11}$$

Proof: The recovery sets $\mathcal{R}_1, \ldots, \mathcal{R}_k \subseteq [m]$ are indices of the disjoint sets of columns of G that add to $\boldsymbol{e}_i$. ∎

It is easy to see that the converse of Lemma 1 is also true. That is, if G is an $s \times m$ binary matrix and $\boldsymbol{c} = \boldsymbol{x}G$, then $x_i = \sum_{j \in \mathcal{R}} c_j$ for all $\boldsymbol{x} = \{0,1\}^s$ if and only if the columns of G indexed by $\mathcal{R} \subseteq [m]$ add to $\boldsymbol{e}_i$. Consequently, an $s \times m$ matrix G is a k-server PIR matrix if and only if (11) holds for all i.

For much more on k-server PIR codes, see the next two sections. In the meantime, the following theorem shows how such codes can be used to construct coded PIR schemes with reduced storage overhead.

Theorem 1: Suppose there exists a k-server PIR code $\mathbb{C}$ of length m and dimension s and a k-server linear PIR protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$. Then there exists a coded PIR scheme with s parts and m shares along with the corresponding coded PIR protocol $\mathcal{P}^*(\mathcal{Q}^*, \mathcal{A}^*, \mathcal{C}^*)$.

Proof: Let G be a generator matrix for $\mathbb{C}$ with property A_k. Then, as in (2) and (6), the coded shares are computed from the s database parts $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s$ as follows:

$$(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m) \;=\; (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s)G \tag{12}$$

Assume Alice wishes to read the i-th bit from the ℓ-th part, namely the bit $x_{\ell,i}$ for some $i \in [n/s]$. First, she invokes the query algorithm $\mathcal{Q}$ to generate k randomized queries: $(q_1, \ldots, q_k) = \mathcal{Q}(k, n/s; i)$. By Lemma 1, there exist k disjoint sets $\mathcal{R}_1, \ldots, \mathcal{R}_k \subseteq [m]$ such that

$$\boldsymbol{x}_\ell \;\coloneqq\; \sum\nolimits_{j \in \mathcal{R}_1} \boldsymbol{c}_j \;=\; \cdots \;=\; \sum\nolimits_{j \in \mathcal{R}_k} \boldsymbol{c}_j \tag{13}$$

The queries $\mathcal{Q}^*(m, s, n; i) = (q_1^*, \ldots, q_m^*)$ that Alice sends to the servers $\mathcal{S}_1, \ldots, \mathcal{S}_m$ are defined as follows. For those indices $j \in [m]$ that do not belong to $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_k$, we can choose q_j^* arbitrarily, since answers from the corresponding servers will be ignored. For $j \in (\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_k)$, we find the unique $t \in [k]$ such that $j \in \mathcal{R}_t$ and set $q_j^* = q_t$. With this setting of $q_1^*, \ldots, q_m^*$, it is easy to see that the privacy guarantees of $\mathcal{Q}$ are inherited by the new query algorithm $\mathcal{Q}^*$. Alice next collects the answers

$$\begin{aligned} a_j \;&=\; \mathcal{A}^*(s, m, j; \boldsymbol{c}_j, q_j^*) \\ &\overset{\text{def}}{=}\; \big\{\mathcal{A}(k, 1; \boldsymbol{c}_j, q_j^*), \ldots, \mathcal{A}(k, k; \boldsymbol{c}_j, q_j^*)\big\} \\ &\qquad \text{for } j \in (\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_k) \end{aligned} \tag{14}$$

where we have assumed (for the time being) that the servers return all the possible answers according to the original answer algorithm $\mathcal{A}$, as in (5). Alice, however, knows the unique index $t \in [k]$ such that $j \in \mathcal{R}_t$, and will retain only $\mathcal{A}(k, t; \boldsymbol{c}_j, q_j^*)$ from the answer in (14). She can now compute

$$a'_t \overset{\text{def}}{=} \sum_{j \in \mathcal{R}_t} \mathcal{A}\big(k, t; \boldsymbol{c}_j, q_j^*\big) \;\coloneqq\; \sum_{j \in \mathcal{R}_t} \mathcal{A}\big(k, t; \boldsymbol{c}_j, q_t\big)$$

$$\coloneqq \mathcal{A}\left(k, t; \sum_{j \in \mathcal{R}_t} \boldsymbol{c}_j, q_t\right) \;\coloneqq\; \mathcal{A}(k, t; \boldsymbol{x}_\ell, q_t) \text{for } t = 1, 2, \ldots, k$$

where the first equality follows from the way queries are set by $\mathcal{Q}^*$, the second equality follows from the linearity of $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$, and the third equality follows from (13). Alice can now complete the retrieval by invoking

$$\begin{aligned} &\mathcal{C}^*(m, s, n;\; i, a_1, \ldots, a_m) \\ &\overset{\text{def}}{=} \mathcal{C}\big(k, n/s;\; i, a'_1, \ldots, a'_k\big) \\ &= \mathcal{C}\big(k, n/s;\; i, \mathcal{A}(k, 1; \boldsymbol{x}_\ell, q_1), \ldots, \mathcal{A}(k, k; \boldsymbol{x}_\ell, q_k)\big) \;\coloneqq\; x_{\ell,i}. \end{aligned}$$

∎

If a coded PIR protocol $\mathcal{P}^*$ is obtained from a linear PIR protocol $\mathcal{P}$ as in the proof of Theorem 1, we will say that $\mathcal{P}^*$ *emulates* $\mathcal{P}$. Let us now assess the communication complexity of such emulation. On the upload side, we see that the number of queries sent by Alice increases from k to m, although each query is slightly shorter since it is generated by $\mathcal{Q}(k, n/s; i)$ rather than $\mathcal{Q}(k, n; i)$. On the download side, the number of answers sent to Alice also increases from k to m. But now each answer is about k times longer, as explained in (5) and (14).

The following theorem addresses this deficiency. In order to state this theorem concisely, let us introduce some notation. Given a k-server PIR protocol $\mathcal{P}$, let $U(\mathcal{P}; n)$ and $D(\mathcal{P}; n)$ denote the worst-case total number of bits uploaded and downloaded, respectively, by $\mathcal{P}$ for a database of length n. For a coded PIR protocol $\mathcal{P}^*$, we define the quantities $U(\mathcal{P}^*; n)$ and $D(\mathcal{P}^*; n)$ in the same way.

Theorem 2: Suppose there exists a k-server PIR code $\mathbb{C}$ of length m and dimension s. Then a linear k-server PIR protocol $\mathcal{P}$ can be emulated by a coded PIR protocol $\mathcal{P}^*$ with s parts and m shares, having communication complexity

$$U\big(\mathcal{P}^*; n\big) \le \frac{m}{k}\, U(\mathcal{P}; n/s) \;+\; m \log_2 k$$

$$\text{and } D\big(\mathcal{P}^*; n\big) \le \frac{m}{k}\, D(\mathcal{P}; n/s).$$

Proof: Based on the foregoing discussion, it would suffice to show how to reduce the number of bits in the answer of (14) by a factor of k. To this end, we introduce the following modification in the proof of Theorem 2. Upon generating

the k randomized queries $(q_1, \ldots, q_k) = \mathcal{Q}(k, n/s;\, i)$, Alice also selects a uniformly random permutation π on $[k]$. As before, the queries q_j^* for those indices j that do not belong to $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_k$ can be set arbitrarily. Also as before, for each $j \in (\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_k)$, Alice finds the unique $t \in [k]$ such that $j \in \mathcal{R}_t$. But now she sets the corresponding query as $q_j^* = q_{\pi(t)}$ and, along with q_j^*, she also *sends the index* $\pi(t)$ to the j-th server $\mathcal{S}_j$. Note that $\pi(t)$ does not reveal any information about either i or ℓ, since it is uniformly random over $[k]$ by the choice of π. However, the server $\mathcal{S}_j$ can now respond with

$$\begin{aligned} a_j &= \mathcal{A}^*\big(s, m, j; \boldsymbol{c}_j, q_j^*\big) \stackrel{\text{def}}{=} \mathcal{A}\big(k, \pi(t);\, \boldsymbol{c}_j, q_j^*\big) \\ &= \mathcal{A}\big(k, \pi(t);\, \boldsymbol{c}_j, q_{\pi(t)}\big). \end{aligned}$$

Using these responses, Alice can compute

$$\begin{aligned} a_t^* &\stackrel{\text{def}}{=} \sum_{j \in \mathcal{R}_t} \mathcal{A}\big(k, \pi(t);\, \boldsymbol{c}_j, q_{\pi(t)}\big) \\ &\coloneqq \mathcal{A}\left(k, \pi(t);\, \sum_{j \in \mathcal{R}_t} \boldsymbol{c}_j, q_{\pi(t)}\right) \coloneqq \mathcal{A}\big(k, \pi(t);\, \boldsymbol{x}_\ell, q_{\pi(t)}\big) \\ &\quad \text{for } t = 1, 2, \ldots, k \end{aligned}$$

Finally, she can invert the permutation by setting $a'_t = a^*_{\pi^{-1}(t)} = \mathcal{A}(k, t; \boldsymbol{x}_\ell, q_t)$, and proceed with retrieving the bit $x_{\ell,i}$ as before. That is, $x_{\ell,i} = \mathcal{C}(k, n/s;\, i, a'_1, \ldots, a'_k)$. ∎

IV. Constructions of k-Server PIR Codes

As we have seen in the previous section, our construction of coded PIR schemes hinges upon the availability of a k-server PIR code, for the appropriate values of k and s. Moreover, the storage overhead of such coded PIR schemes is inversely proportional to the rate of the underlying code. In this section, we present several constructions of k-server PIR codes of high rate. We shall see how such codes can be constructed from Steiner systems, from majority-logic decodable codes, from bipartite graphs of girth 6, from constant-weight codes, and from subset codes. We shall also see that k-server PIR codes are closely related to batch codes [21], [31], [41], to locally recoverable codes with availability [19], [20], [29], [30], [38], and to other kinds of combinatorial objects.

To aid the discussion in this section, let us introduce the following functions. For all $s \geq 1$ and $k \geq 2$, we define $M(s, k)$ and $\rho(s, k)$ as *the smallest possible length* and *the lowest possible redundancy*, respectively, of a binary k-server PIR code of dimension s. The storage overhead of our coded PIR schemes is then given by

$$\text{storage overhead with } s \text{ parts} = \frac{M(s,k)}{s} = 1 + \frac{\rho(s,k)}{s} \tag{15}$$

We observe that the problem of determining $M(s, k)$ and $\rho(s, k)$ is trivial for $k = 2$. Let $\mathbb{C}$ be the $(s+1, s)$ parity code, obtained by appending an overall parity bit to a message $\boldsymbol{x} \in \{0, 1\}^s$. Then it is easy to see that *any* generator matrix for $\mathbb{C}$ has property A_2, and so $M(s, 2) = s + 1$. Thus, in the remainder of this section, we focus on $k \geq 3$.

A. The Cubic Construction

Our first construction is based on the geometry of multidimensional cubes. This approach is similar to the one used in [21] to construct certain types of batch codes. We point out, however, that the construction in [21] is recursive, while ours is not. We begin by illustrating the general idea with the help of a simple example.

Example 3: Suppose that $k = 3$ and $s = \sigma^2$ for a positive integer σ. We arrange the σ^2 message bits in the form of a $\sigma \times \sigma$ square. That is, given a message $\boldsymbol{x} \in \{0, 1\}^s$, we index its σ^2 bits as $x_{i,j}$ for $i, j \in [\sigma]$. To each message, we append 2σ parity bits $c_1, \ldots, c_\sigma$ and $c'_1, \ldots, c'_\sigma$ defined as follows:

$$\begin{aligned} c_i &= x_{i,1} + x_{i,2} + \cdots + x_{i,\sigma} \text{ and} \\ c'_j &= x_{1,j} + x_{2,j} + \cdots + x_{\sigma,j} \text{ for } i, j \in [\sigma] \end{aligned}$$

These bits are simply parities on the rows and columns of the $\sigma \times \sigma$ message array. The resulting code, which we denote by $\mathcal{C}(\sigma, 3)$, is very similar to the product of two single-parity-check codes, except that it is shorter by one bit (we removed a corner of the $(\sigma+1) \times (\sigma+1)$ coded array, since we do not need it). Note that this code is *systematic*, meaning that the first s coded bits are the message bits themselves. One can easily verify that, given a codeword of $\mathcal{C}(\sigma, 3)$, every message bit can be recovered in $k = 3$ different ways: as itself, from its row parity, and from its column parity. Explicitly, for all $i, j \in [\sigma]$, we have

$$\begin{aligned} x_{i,j} &= x_{i,1} + \cdots + x_{i,j-1} + c_i + x_{i,j+1} + \cdots + x_{i,\sigma} \\ &= x_{1,j} + \cdots + x_{i-1,j} + c'_j + x_{i+1,j} + \cdots + x_{\sigma,j} \end{aligned} \tag{16}$$

and the corresponding sets of indices are disjoint. It follows that $\mathcal{C}(\sigma, 3)$ is a 3-server PIR code. Since its length is $\sigma^2 + 2\sigma$, we have $M(\sigma^2, 3) \leq \sigma^2 + 2\sigma$. More generally, if we take a similar product of two systematic k-server PIR codes, we obtain $\rho(\sigma^2, 2k-1) \leq 2\sigma\rho(\sigma, k)$; what we have here is the special case $\rho(\sigma^2, 3) \leq 2\sigma\rho(\sigma, 2) = 2\sigma$. ∎

Let us now extend the construction of Example 3 from two to $k-1$ dimensions, where $k \geq 4$ is arbitrary. We start by defining σ as the least integer such that $s \leq \sigma^{k-1}$. Given a message $\boldsymbol{x} \in \{0, 1\}^s$, we extend it to length σ^{k-1} by appending zeros, if necessary. Next, we arrange the message bits in the form of a $(k-1)$-dimensional cube of side σ — that is, we index these bits as $x_{i_1, i_2, \ldots, i_{k-1}}$, with $i_1, i_2, \ldots, i_{k-1} \in [\sigma]$. To each message, we append $(k-1)\sigma^{k-2}$ parity bits defined as follows:

$$\begin{aligned} c^{(1)}_{i_2, i_3, \ldots, i_{k-1}} &= x_{1, i_2, \ldots, i_{k-1}} + x_{2, i_2, \ldots, i_{k-1}} + \cdots + x_{\sigma, i_2, \ldots, i_{k-1}} \\ &\quad \text{for } i_2, i_3, \ldots, i_{k-1} \in [\sigma] \end{aligned} \tag{17}$$

$$\begin{aligned} c^{(2)}_{i_1, i_3, \ldots, i_{k-1}} &= x_{i_1, 1, \ldots, i_{k-1}} + x_{i_1, 2, \ldots, i_{k-1}} + \cdots + x_{i_1, \sigma, \ldots, i_{k-1}} \\ &\quad \text{for } i_1, i_3, \ldots, i_{k-1} \in [\sigma] \end{aligned} \tag{18}$$

$$\vdots$$

$$\begin{aligned} c^{(k-1)}_{i_1, i_2, \ldots, i_{k-2}} &= x_{i_1, i_2, \ldots, 1} + x_{i_1, i_2, \ldots, 2} + \cdots + x_{i_1, i_2, \ldots, \sigma} \\ &\quad \text{for } i_1, i_2, \ldots, i_{k-2} \in [\sigma] \end{aligned} \tag{19}$$

We then puncture from the resulting codeword those zero bits, if any, that were originally used to extend the message to length σ^{k-1}. Let $\mathcal{C}(\sigma, k)$ denote the systematic linear

code defined by this encoding process. The following theorem proves that it is a k-server PIR code.

Theorem 3: For $s \geq 1$ and $k \geq 3$, let σ denote the unique positive integer such that $(\sigma-1)^{k-1} < s \leq \sigma^{k-1}$. Then

$$M(s,k) \;\leq\; s + (k-1)\sigma^{k-2} \tag{20}$$

Proof: Consider the code $\mathcal{C}(\sigma,k)$ defined by (17)–(19). It is obvious that this is a code of dimension s and length $s+(k-1)\sigma^{k-2}$. With the indexing scheme of (17)–(19), for every message bit $x_{i_1,i_2,\ldots,i_{k-1}}$, we have

$$\begin{aligned} x_{i_1,i_2,\ldots,i_{k-1}} &= c^{(1)}_{i_2,i_3,\ldots,i_{k-1}} + \sum_{j=1 j\neq i_1}^{\sigma} x_{j,i_2,\ldots,i_{k-1}} \\ &= c^{(2)}_{i_1,i_3,\ldots,i_{k-1}} + \sum_{j=1 j\neq i_2}^{\sigma} x_{i_1,j,\ldots,i_{k-1}} \\ &= \cdots = c^{(k-1)}_{i_1,i_2,\ldots,i_{k-2}} + \sum_{j=1 j\neq i_{k-1}}^{\sigma x} i_1,i_2,\ldots,j. \end{aligned}$$

It is easy to see that the corresponding sets of indices are disjoint, as in (16). Hence, for every message bit, we have k disjoint recovery sets, which implies that $\mathcal{C}(\sigma,k)$ is a k-server PIR code. ∎

Corollary 1: For all fixed $k \geq 2$, there exist k-server coded PIR schemes whose storage overhead approaches 1 as the number of parts s grows.

Proof: By Theorem 1, we can construct a k-server coded PIR scheme with storage overhead m/s whenever there exists a k-server PIR code of length m and dimension s. Thus it would suffice to show that $\lim_{s\to\infty} M(s,k)/s = 1$ for all fixed k. But this follows immediately from Theorem 3. For simplicity, let us assume that $s = \sigma^{k-1}$ (the difference between $(\sigma-1)^{k-1}$ and σ^{k-1} becomes negligible in the limit as $s \to \infty$). Then (20) implies that

$$\frac{M(s,k)}{s} \;\leq\; 1 + \frac{(k-1)\,s^{\frac{k-2}{k-1}}}{s} \;=\; 1 + \frac{k-1}{\sqrt[k-1]{s}} \;\xrightarrow{s\to\infty}\; 1.$$

∎

One conclusion from Theorem 3 and Corollary 1 is that it is *easy* to bring the asymptotic storage overhead of coded PIR schemes down to 1 as $s \to \infty$. Our very first construction already achieves this goal for all fixed k. Therefore, we shall henceforth focus on *how fast* the storage overhead approaches 1 as the number of parts grows. To this end, it will be more convenient to consider the redundancy function $\rho(s,k)$. The cubic construction of (17)–(19) shows that

$$\rho(s,k) \;\leq\; (k-1)\lceil \sqrt[k-1]{s}\,\rceil^{k-2} \tag{21}$$

As we shall see later in this section, this bound is far from optimal, especially for $k \geq 4$. Even for $k = 3$, there is only one case where the cubic construction achieves the optimal redundancy, namely $\rho(4,3) = 4$.

B. One-Step Majority Logic Codes

One-step majority logic decoding is a method to perform fast decoding by looking at disjoint parity check constraints that only intersect on a single bit (see [13, Ch. 8]) These parity check constraints correspond to the codewords in the dual code, and hence, for a linear code $[n,k,d]$, the goal is to find, for each $i \in [n]$, a set of codewords in the dual code that intersect only on the i-th bit. These codewords are said to be *orthogonal* on the i-th bit. The maximum number of such orthogonal vectors in the dual code (for every bit) is denoted by J, and if $J = d-1$, then the code is called *completely orthogonalizable*.

In other words, if an $[n,k]$ code has J orthogonal vectors on the i-th coordinate for some $i \in [n]$, then its dual code $\mathcal{C}^{\perp}$ has $k-1 = J$ codewords that are orthogonal on coordinate i. Assume that these codewords are given by

$$c^{\perp}_j = x_i + x_{j_1} + x_{j_2} + \cdots + x_{j_{p_j}} \qquad \forall 1 \leq j \in J, \tag{22}$$

where the sets $\{i\}$, and $\{j_1, j_2, \ldots, j_{p_i}\}$ for $j \in [J]$ are mutually disjoint. Such $[n,k,d]$ code with J orthogonal vectors for each $i \in [n]$ is called a *one-step majority logic code with J orthogonal vectors*. Note that the definition of one-step majority logic codes is almost identical to the one of PIR codes given in Definition 5. While one-step majority logic codes guarantee that orthogonal vectors (or mutually disjoint sets) exist for all the bits in the code, in PIR codes we require this property only for the s information bits. While it is not always straightforward to construct an appropriate generator matrix from a given code such that the k-server PIR property holds, for the case of one-step majority logic codes, we can always pick a systematic generator matrix and hence the PIR property follows. Lastly we note that the idea of using one-step majority logic codes was motivated by the recent work on codes for locality and availability in [20]. We demonstrate the construction of such codes in the following example.

Example 4: Consider a $(15,7)$ cyclic code generated by the polynomial $g(x) = 1 + x^4 + x^6 + x^7 + x^8$. The parity-check matrix of this code in the systematic form is given by

$$H \;=\; \begin{bmatrix} 1&0&0&0&0&0&0&0&1&1&0&1&0&0&0 \\ 0&1&0&0&0&0&0&0&0&1&1&0&1&0&0 \\ 0&0&1&0&0&0&0&0&0&0&1&1&0&1&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&1&0&1 \\ 0&0&0&0&1&0&0&0&1&1&0&1&1&1&0 \\ 0&0&0&0&0&1&0&0&0&1&1&0&1&1&1 \\ 0&0&0&0&0&0&1&0&1&1&1&0&0&1&1 \\ 0&0&0&0&0&0&0&1&1&0&1&0&0&0&1 \end{bmatrix}$$

We observe that the following codewords in $C^{\perp}$

$$\begin{aligned} h_3 &= (0\;0\;0\;1\;0\;0\;0\;0\;0\;0\;0\;1\;1\;0\;1), \\ h_{1+5} &= (0\;1\;0\;0\;0\;1\;0\;0\;0\;0\;0\;0\;0\;1\;1), \\ h_{0+2+6} &= (1\;0\;1\;0\;0\;0\;1\;0\;0\;0\;0\;0\;0\;0\;1), \\ h_7 &= (0\;0\;0\;0\;0\;0\;0\;1\;1\;0\;1\;0\;0\;0\;1), \end{aligned}$$

are orthogonal on coordinate 14. That gives us five mutually disjoint sets $\{3,11,12\}$, $\{1,5,13\}$, $\{0,2,6\}$, $\{7,8,10\}$, and $\{14\}$ that are required in Definition 5 to make five different queries on server 14. The same statement is correct for all other coordinates due to the cyclicity of the code. So, $\mathcal{C}$ is a 5-server PIR code. The storage overhead of the coded PIR scheme based on $\mathcal{C}$ is given by $\frac{1}{\text{code rate}} = \frac{15}{7}$, which is significantly better than the uncoded PIR.

There are several algebraic constructions for one-step majority logic codes. However, the explicit relation between the code length and redundancy is only known for a few of them. *Type-1 Doubly Transitive Invariant (DTI) Codes* (see [13, p. 289]) are cyclic codes with *almost* completely orthogonalizable property. An explicit relation between the code length $m = (2^M - 1)$, code dimension (s), and the number of orthogonal codewords in the dual code (J), is known for specific choices of these parameters:

- Case I. Let θ, ℓ be two positive integers. For $M = 2\theta\ell$ and $J = 2^\ell + 1$, the redundancy of the type-1 DTI code of length m is given by
$$r = \left(2^{\theta+1} - 1\right)^\ell - 1. \tag{23}$$
We refer to these codes by $\mathcal{C}_{C_1}(\theta, \ell)$.
- Case II. Let λ, ℓ be two positive integers. For $M = \lambda\ell$ and $J = 2^\ell - 1$, the redundancy of the type-1 DTI code of length m is given by
$$r = 2^M - \left(2^\lambda - 1\right)^\ell - 1. \tag{24}$$
We refer to these codes by $\mathcal{C}_{C_2}(\lambda, \ell)$.

We refer the reader to [13] for the algebraic construction and the calculation method used in deriving these parameters.

Theorem 4: For any positive integers θ, ℓ, and λ, the Type-1 DTI codes $\mathcal{C}_{C_1}(\theta, \ell)$, $\mathcal{C}_{C_2}(\lambda, \ell)$ are $(2^\ell + 2)$-server, and 2^ℓ-server PIR codes, respectively. In particular, we get that
$$M\left(2^{2\theta\ell} - \left(2^{\theta+1} - 1\right)^\ell, 2^\ell + 2\right) \le 2^{2\theta\ell} - 1,$$
$$M\left(\left(2^\lambda - 1\right)^\ell - 1, 2^\ell\right) \le 2^{\lambda\ell} - 1,$$
and hence for any fixed k, there exists a family of k-server PIR codes with asymptotic storage overhead of $1 + \mathcal{O}(s^{-\frac{1}{2}})$.

Proof: We have already shown that a one-step majority logic code with J orthogonal vectors, is also a $(J+1)$-server PIR code. So we are left with only calculating the code dimensions according to the redundancies in (23) and (24).

- For $\mathcal{C}_{C_1}(\theta, \ell)$, the code dimension is given by $s = m - r = 2^{2\theta\ell} - (2^{\theta+1} - 1)^\ell$.
- For $\mathcal{C}_{C_2}(\lambda, \ell)$, the code dimension is given by $s = m - r = (2^\lambda - 1)^\ell - 1$.

So the upper bounds are validated. For the asymptotic analysis, we point out that for a given fixed J, as the number of servers grows, the rates of the codes in both cases I, and II become arbitrary close to 1. In particular, when k is fixed and s becomes large, the storage overhead in $\mathcal{C}_{C_2}(\lambda, \ell)$ is
$$\frac{2^{\lambda\ell} - 1}{\left(2^\lambda - 1\right)^\ell - 1} \approx \left(\frac{2^\lambda}{2^\lambda - 1}\right)^\ell \approx 1 + \frac{\ell}{2^\lambda - 1} = 1 + O\left(s^{-\frac{1}{\ell}}\right),$$
which is an improvement compared to Theorem 3 in the asymptotic regime. An even better storage overhead is achieved by $\mathcal{C}_{C_1}(\theta, \ell)$ codes in the asymptotic regime:
$$\frac{2^{2\theta\ell} - 1}{2^{2\theta\ell} - \left(2^{\theta+1} - 1\right)^\ell} = 1 + O\left(s^{-\frac{1}{2}}\right). \tag{25}$$
■

Note that this construction not only outperforms the former ones with respect to the upper bound on the asymptotic storage overhead, but also gives a bound on $M(s, k)$ that does not depend on k is in the asymptotic regime. Considering that the construction based on Steiner systems also result in a similar bound, we ask the following two questions regarding the asymptotic storage overhead behavior of k-server PIR codes.

C. PIR Codes Based on Steiner Systems

The idea behind a construction of any k-server PIR code $\mathcal{C}$ is to form, for every information bit, k mutually disjoint subsets of $[m]$, such that the information bit can be recovered by a linear combination of the bits in each set. Assume that $\mathcal{C}$ is a systematic $[m, m-r=s]$ k-server PIR code. Then, we can partition its bits into two parts; the first one consists of the s information bits, denoted by $x_1, \ldots, x_s$ and the second one is the r redundancy bits $p_1, \ldots, p_r$, where every redundancy bit p_i is characterized by a subset $S_i \subseteq [s]$ such that $p_i = \sum_{j \in S_i} x_j$.

According to this representation of systematic codes, every collection $\mathcal{S} = (S_1, \ldots, S_r)$ of subsets of $[s]$ defines a systematic $[s + r, s]$ linear code $\mathcal{C}_B(\mathcal{S})$. In the next lemma, we give sufficient (but not necessary) conditions such that the code $\mathcal{C}_B(\mathcal{S})$ is a k-server PIR code.

Lemma 2: Let $\mathcal{S} = (S_1, \ldots, S_r)$ be a collection of subsets of $[s]$, such that

1) For all $i \in [s]$, i appears in at least $k-1$ subsets,
2) For all $j, \ell \in [r]$, $|S_j \cap S_\ell| \le 1$.

Then, $\mathcal{C}_B(\mathcal{S})$ is a k-server PIR code.

Proof: For any information bit x_i, $i \in [s]$, according to the first condition there exist some $k-1$ subsets $S_{i_1}, \ldots, S_{i_{k-1}}$, such that $i \in S_{i_j}$ for $j \in [k-1]$. For each $j \in [k-1]$, let $\mathcal{R}_j$ be the set $\mathcal{R}_j = \{x_\ell \ \ \ell \in S_{i_j}, \ell \ne i\} \cup \{p_{i_j}\}$, and finally let $\mathcal{R}_k = \{x_i\}$. According to the second condition all these k sets are mutually disjoint. Finally, it is straightforward to verify that x_i is the sum of the bits in every set, and thus $\mathcal{C}_B(\mathcal{S})$ is a k-server PIR code. ■

After determining the conditions in which the code $\mathcal{C}_B(\mathcal{S})$ is a k-server PIR code, we are left with the problem of finding such collections of subsets. Our approach to fulfill the conditions stated in Lemma 2 is to search for existing combinatorial objects in the literature. One such an object is a *Steiner system*. A Steiner system with parameters t, ℓ, n, denoted by $S(t, \ell, n)$, is an n-element set S together with a set of ℓ-element subsets of S (called blocks) with the property that each t-element subset of S is contained in exactly one block. It is also commonly known that the number of subsets in a Steiner system $S(t, \ell, n)$ is $\binom{n}{t}/\binom{\ell}{t}$ and every element is contained in exactly $\binom{n-1}{t-1}/\binom{\ell-1}{t-1}$ subsets.

In order to satisfy the conditions in Lemma 2, we chose Steiner systems with $t = 2$ so the intersection of every two subsets contains at most one element. Furthermore, in a Steiner system $S(2, \ell, n)$, the number of subsets is $\binom{n}{2}/\binom{\ell}{2} = n(n-1)/\ell(\ell-1)$ and every element is contained in $(n-1)/(\ell-1)$ subsets. Thus, we conclude with the following theorem.

Theorem 5: If a Steiner system $S(2, \frac{s-1}{k-1} + 1, s)$ exists, then there exists an $[m = s + r, s]$ k-server PIR code where $r = \frac{s(k-1)^2}{s+k-2}$. Thus, under this assumption we have
$$M(s, k) \le s + \frac{s(k-1)^2}{s + k - 2}. \tag{26}$$

Moreover, if a Steiner system $S(2, k-1, r)$ exists, then we have a k-server PIR code with parameters $[m, s] = [r + \frac{r(r-1)}{(k-1)(k-2)}, \frac{r(r-1)}{(k-1)(k-2)}]$. Thus,

$$M\left(\frac{r(r-1)}{(k-1)(k-2)}, k\right) \leq r + \frac{r(r-1)}{(k-1)(k-2)}. \tag{27}$$

Proof: Let $\mathcal{S}$ be a Steiner system $S(2, \frac{s-1}{k-1} + 1, s)$, so the number of subsets in $\mathcal{S}$ is

$$r = \frac{s(s-1)}{\left(\frac{s-1}{k-1} + 1\right)\frac{s-1}{k-1}} = \frac{s(k-1)^2}{s+k-2},$$

and every element is contained in

$$\frac{s-1}{(s-1)/(k-1)} = k-1$$

subsets. We also have that the intersection of every two subsets contains at most one element, so the conditions in Lemma 2 hold and $\mathcal{C}_B(\mathcal{S})$ is a k-server PIR code. To prove the bound given in (27), let $\tau = \binom{r}{2}/\binom{k-1}{2}$ be the number of $(k-1)$-element subsets of $S(2, k-1, r)$, and denote them by $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_\tau \subset [r]$. Let us construct the dual Steiner system $S'(2, \frac{r-1}{k-2}, \tau)$ which consists of r $(\frac{r-1}{k-2})$-element subsets of $[\tau]$ denoted by $\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_r$, and has the property that $\mathcal{S}'_i = \{a | a \in [\tau], i \in \mathcal{S}_a\}$. We now use the first statement in (26) to construct the code $\mathcal{C}_B(S')$. It is clear that the redundancy of $\mathcal{C}_B(S')$ is given by r, and the code length is given by $r + \tau = r + \frac{r(r-1)}{(k-1)(k-2)}$. ∎

Example 5: A finite projective plane of order q, with the lines as blocks, is an $S(2, q+1, q^2+q+1)$ Steiner system. Since $q + 1 = \frac{(q^2+q+1)-1}{(q+2)-1} + 1$, we conclude that there exists an $[s+r, s]$ $(q+2)$-server PIR code, with $s = q^2 + q + 1$ information bits and

$$r = \frac{(q^2+q+1)(q+1)^2}{q^2+q+1+q+2-2} = \frac{(q^2+q+1)(q+1)^2}{(q+1)^2} = q^2+q+1$$

redundancy bits. Note that the storage overhead of this code is 2.

In order to evaluate the bound (27) in Theorem 5, one is required to figure out the existence of $S(t, \ell, n)$ in general. Indeed, Wilson's Theorem [43] claims that for a fixed ℓ, and sufficiently large n, a Steiner system $S(2, \ell, n)$ exists given that the following two conditions (also known as divisibility conditions) are satisfied: $\ell - 1$ divides $n - 1$ and $\ell(\ell-1)$ divides $n(n-1)$. Wilson Theorem guarantees the existence of $S(2, k-1, r)$ for infinitely many values of r. Hence, for a fixed k, there are arbitrary large values for r such that the bound in (27) holds. Hence, the redundancy behaves asymptotically according to $\rho(s, k) = O(s^{1/2})$, which improves upon the cubic construction.

D. Bipartite Graphs and Constant-Weight Codes

Assume that G is a generator matrix of a systematic k-server PIR code $\mathcal{C}$ of length m and dimension s. We rewrite G as

$$G = [I_s | A_{s \times r}], \tag{28}$$

where I_s is the $s \times s$ identity matrix and $A_{s \times r}$ corresponds to the r parities in $\mathcal{C}$. Let us look at the systematic PIR codes from a graph theory point of view by interpreting A as the incidence matrix of a bipartite graph $\mathcal{G}$ with partite sets $\mathcal{X} = \{x_1, x_2, \ldots, x_s\}$ and $\mathcal{P} = \{p_1, p_2, \ldots, p_r\}$, and edges $\mathcal{E} = \{\{x_i, p_j\} | A_{ij} = 1\}$. We call $\mathcal{C}$ by the *Systematic PIR code based on* $\mathcal{G}$. The following lemma is an equivalent statement to Lemma 2.

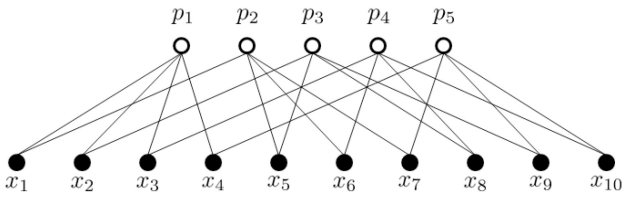

Fig. 2. The bipartite graph $\mathcal{G}$ associated with the matrix $A_{10\times5}$.

Lemma 3: Let $\mathcal{G}$ be a bipartite graph with partite sets $\mathcal{X} = \{x_1, x_2, \ldots, x_s\}$, $\mathcal{P} = \{p_1, p_2, \ldots, p_r\}$, and the incidence matrix A, where $k - 1 = \min_{x \in \mathcal{X}} \deg(x)$. Further, assume that $\mathcal{G}$ has no cycles of length 4. If $\mathcal{C}$ is the systematic code based on $\mathcal{G}$ with generator matrix defined in (28), then $\mathcal{C}$ is a k-server PIR code of length $m = s + r$ and dimension s.

Proof: Consider x_i and $k-1$ of its parity neighbors $\{p_{i_1}, p_{i_2}, \ldots, p_{i_{k-1}}\} \subset \mathcal{P}$. Let $\mathcal{R}_i^j \subset \mathcal{X}$ denote the neighbor set of p_{i_j}. Since $\mathcal{G}$ is 4-cycle free, the sets $\mathcal{R}_i^j \setminus \{x_i\}$ (for a fixed i and $j \in [k-1]$) are mutually disjoint. It is also easy to see that $\{x_i\}$, and $\{p_{i_j}\} \cup \mathcal{R}_i^j \setminus \{x_i\}$ (for $j = 1, 2, \ldots, k-1$) form k disjoint recovery sets for u_i. In other words,

$$p_{i_j} = \sum_{x_\alpha \in \mathcal{R}_i^j} x_\alpha \Rightarrow x_i = p_{i_j} + \sum_{x_\alpha \in \mathcal{R}_i^j \setminus \{x_i\}} x_\alpha \quad \text{for } j = 1, 2, \ldots, k-1.$$ ∎

Now we are ready to proceed to the final construction of k-server PIR codes, which will be first demonstrated by an example.

Example 6: Consider the 3-server PIR code $\mathcal{C}$ given by the systematic generator matrix

$$G = [I_{10} | A_{10\times5}] = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&1&0&1&0&0 \\ 0&0&1&0&0&0&0&0&0&0&1&0&0&1&0 \\ 0&0&0&1&0&0&0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&0&0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&1&0&1&0 \\ 0&0&0&0&0&0&1&0&0&0&0&1&0&0&1 \\ 0&0&0&0&0&0&0&1&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&1&1 \end{bmatrix}.$$

The corresponding bipartite graph is also shown in Fig. 2.

We observe that $\deg(x_i) = 2$ for all i as expected since we only need $k - 1 = 2$ elements in $\mathcal{N}(x_i)$ to recover x_i, where $\mathcal{N}(\alpha)$ is the neighborhood set of α. Moreover,

$$|\mathcal{N}(x_i) \cap \mathcal{N}(x_j)| \leq 1 \text{ for } i \neq j, \tag{29}$$

which guarantees that the recovering sets for x_i are mutually disjoint.

The requirement that $\deg(x_i) \geq k - 1$ can be replaced with $\deg(x_i) = k - 1$ in this construction. This motivates us to look at constant-weight codes where the codewords are all rows in the matrix A. For instance, we look at a constant weight code

with weight $k-1$ and minimum distance $2k-4$. Let $A(k,r)$ be the list of the largest code of length r whose codewords have weight $k-1$ and their minimum distance is $2k-4$. $\mathcal{C}_D(k,r)$ is a k-server PIR code defined by its systematic generator matrix $G_{(k,r)} = [I|A(k,r)]$.

We use the notation $B(n,w,d)$ to denote the maximum number of codewords of length n and weight w with minimum distance d. There are numerous works and studies aiming to determine the precise values of $B(n,w,d)$ in general, but the explicit formula is only found for the trivial cases. A complete collection of the known precise values and both upper bounds and lower bounds on $B(n,w,d)$ is given in [6].

Theorem 6: For any k the code $\mathcal{C}_D(k,r)$ is a k-server PIR code. In particular, we get that for any positive integer k

$$M\big(B(r,k-1,2k-4),k\big) \le B(r,k-1,2k-4)+r.$$

Proof: Let $\mathcal{G}$ be the bipartite graph whose incidence matrix is $M(k,r)$. Clearly, $\deg(x) = k-1$ for all $x \in \mathcal{X}$. Also, $|\mathcal{N}(x)| = |\mathcal{N}(y)| = k-1$, and $|(\mathcal{N}(x) \cup \mathcal{N}(y)) \setminus (\mathcal{N}(x) \cap \mathcal{N}(y))| \ge 2k-4$, which provides that $\mathcal{N}(x) \cap \mathcal{N}(y)| \le 1$. Hence, all of the conditions in Lemma 3 are satisfied and $\mathcal{C}_D(k,r)$ is a k-server PIR code. To validate the parameters in the theorem it suffices to note that $|A(k,r)| = B(r,k-1,2k-4)$, so we have $B(r,k-1,2k-4)$ rows in G, and r, the length of the codewords in $A(k,r)$, determines the redundancy. ∎

Example 7: The only known explicit formula for $B(n,w,d)$ is when $w=2$ and $d=2$. It is easy to see that any two different codewords of weight 2 have distance at least 2 as well. Hence $B(n,2,2) = \binom{n}{2}$. So,

$$M\left(\binom{n}{2},3\right) \le n + \binom{n}{2}. \tag{30}$$

According to inequality 30, we observe again that the asymptotic behavior of $M(s,3)-s$ is $O(s^{1/2})$. The construction based on Steiner systems and the last construction based upon constant-weight codes are both equivalent to the problem of finding bipartite graphs with s vertices on the left and r vertices on the right, where all the left vertices have degree k, the graph has girth at least 6, while minimizing the value of r. Clearly, if a Steiner system with the desired parameter exists, then it is an optimal solution. However, constant-weight codes provide a solution in the general case particularly when the desired Steiner system does not exist. The following theorem on bipartite graphs is both well-known and trivial and shows that by using this method of construction for k-server PIR codes, we can not achieve a better asymptotic storage overhead than the one achieved by Steiner systems. We include here the proof for the sake of completeness of the results in the paper.

Theorem 7: Let $\mathcal{G}$ be a bipartite graph with partite sets $\mathcal{X} = \{x_1, x_2, \ldots, x_s\}$ and $\mathcal{P} = \{p_1, p_2, \ldots, p_r\}$, where $\deg(x) = k > 2$ for all $x \in \mathcal{X}$. If the graph has girth at least 6, then $r(r-1) \ge sk(k-1)$. Hence, for a fixed k, we have $r = O(s^{1/2})$.

Proof: Let $\mathcal{N}(x)$ denote the neighbor set of the vertex x. Since the graph has no 4-cycles, then there are no $i,j, \in [s]$ and $a,b \in [r]$, such that

$$\{p_a, p_b\} \subset \mathcal{N}(x_i) \quad \text{and} \quad \{p_a, p_b\} \subset \mathcal{N}\big(x_j\big)$$

Therefore,

$$\binom{r}{2} \ge \sum_{i\in[s]} \binom{|\mathcal{N}(x_i)|}{2} = s\binom{k}{2}.$$ ∎

V. Optimal Storage Overhead for Small s and k

In this section, we study the value of $M(s,k)$ for small s and k. In order to give the best upper bounds, we benefit from a few supplementary lemmas that together with the constructions introduced in Section IV form a recursive method in deriving the upper bounds on $M(s,k)$. Note that the constructions introduced in Section IV do not cover all values of s and k. The following lemmas give simple tools to derive upper bounds for all values of s and k.

Lemma 4: We have the following inequalities for all non-negative integer values of s, k, s', and k':

1) $M(s,k+k') \le M(s,k) + M(s,k')$,
2) $M(s+s',k) \le M(s,k) + M(s',k)$,
3) $M(s,k) \le M(s,k+1) - 1$,
4) $M(s,k) \le M(s+1,k) - 1$.

Proof: To prove the inequality in *(a)*, assume that $\mathcal{C}$ and $\mathcal{C}'$ are k-server and k'-server PIR codes with parameters $[m,s]$ and $[m',s]$, and their generator matrices are given by G and G', respectively. It is easy to see that the concatenation of $\mathcal{C}$ and $\mathcal{C}'$ is a $(k+k')$-server PIR code with parameters $[m+m',s]$ and its generator matrix is given by $G_{\text{conc}} = [G \mid G']$. To prove the inequality in *(b)*, assume again that $\mathcal{C}$ and $\mathcal{C}'$ are k-server PIR codes with parameters $[m,s]$ and $[m',s']$, and their generator matrices are given by G and G', respectively. The direct sum code (also known as the product code) of $\mathcal{C}$ and $\mathcal{C}'$ is a k-server PIR code with parameters $[m+m', s+s']$ whose generator matrix is given by

$$G^* = \begin{bmatrix} G & 0_{s\times m'} \\ 0_{s'\times m} & G' \end{bmatrix}.$$

To prove *(c)*, let us assume that $\mathcal{C}$ is a $(k+1)$-server PIR code with parameters $[m,s]$ and a generator matrix G. According to Definition 5, for every information bit u_i, $i \in [s]$, there exist $k+1$ mutually disjoint sets $\mathcal{R}_{i,1}, \ldots, \mathcal{R}_{i,k+1} \subset [m]$ such that for all $j \in [k]$, u_i is a linear function the bits in $\mathcal{R}_{i,j}$. It is now clear that deleting one of the coordinates from G or equivalently puncturing the code $\mathcal{C}$ in one of its coordinates can truncate at most one of these disjoint recovery sets. Hence the punctured code $\mathcal{C}_{\text{punc}}$ whose parameters are given by $[m-1,s]$ is a k-server PIR code. We postpone the proof of part *(d)* to the end of this section, where we discuss whether Definition 5 is a property of the generator matrix or it can be interpreted as a property of the code itself. ∎

Lemma 5: If k is odd, then $M(s,k+1) = M(s,k)+1$.

Proof: Utilizing part (c) in Lemma 4, it only suffices to show that if k is odd, then $M(s,k+1) \le M(s,k)+1$. To do so, assume that $\mathcal{C}$ is a k-server PIR code with parameters $[m,s]$ and generator matrix G. For any $i \in [s]$ we should be able to find k disjoint subsets of columns where the columns in each subset sum up to the vector $\boldsymbol{e}_i$. If the sum of all columns in G is $\mathbf{0}$, then clearly the sum of the remaining columns (the ones that are left out of the k subsets) is also the vector $\boldsymbol{e}_i$. Hence the code is actually a $(k+1)$-server PIR code and we

are done. If not, append one more column to G so that the sum of all the columns is $\mathbf{0}$. Then the resulting matrix is a generator matrix for a $(k+1)$-server PIR code. ■

By selecting the best constructions for $M(s,k)$ from Section IV for each individual s and k, it is possible to drive upper bounds on $M(s,k)$ for all values of s, and k. We observe that the storage overhead is significantly improved compared to the traditional uncoded PIR scheme. Moreover, the inequality $M(s,k) \geq s+\sqrt{s}$ always hold. The asymptotic behavior of $M(s,k)$ is discussed in next section.

In the remainder of this section, we seek to address the following key question: Is it the generator matrix that has the k-server PIR property, or it can be interpreted as a property of the code? Let us begin the discussion with the following definition.

Definition 6: We say that an $[m, m-s]$ binary linear code C has property B_k if there exist s cosets of C such that:

1) Every coset contains k disjoint vectors, and
2) The linear span of these cosets is the entire space $\mathbb{F}_2^m$.

Theorem 8: If the code C has the property B_k, then its dual code is a k-server PIR code.

Proof: We show that the definition 6 and 5 are equivalent. Clearly, Definition 6 above is a property of a code, not a matrix. Now, given a generator matrix G with for the PIR code C', we get a code C with property B_k by simply taking C to be the code defined by G as its parity-check matrix. Now lets proceed to the other direction.

Assume that a code C with property B_k is given. Let $C_1, C_2, \ldots, C_s$ be the s linearly independent cosets of C, each containing k disjoint vectors. Start with an arbitrary parity-check matrix H for C. Let $\sigma_1, \sigma_2, \ldots, \sigma_s$ denote the syndromes of $C_1, C_2, \ldots, C_s$ with respect to H. Let S be the $s \times s$ matrix having these syndromes as its columns. Note that condition b) of Definition 6 guarantees that S is full-rank. Now form the $s \times (m+s)$ matrix $[H|S]$, and perform elementary row operations on this matrix to get $[H'|S']$ where S' is the $s \times s$ identity matrix. Then the matrix H' is a generator matrix for the k-server PIR code C', which is clearly the dual code of C. ■

The following lemma from the theory of the linear codes is essential for the proof of part *(d)* in Lemma 4. We leave the proof to the reader.

Lemma 6: Let C be an (m,k) binary linear code. Given a positive $t \leq m-k$, let $C_1, C_2, \ldots, C_t$ be cosets of C, and let $s_1, s_2, \ldots, s_t$ be their syndromes. Then

$$\dim\big(\text{span}(C_1, C_2, \ldots, C_t)\big) = k+t$$

if and only if the syndromes $s_1, s_2, \ldots, s_t$ are linearly independent.

We are now able to prove part 4) of Lemma 4 as promised earlier.

Proof of Lemma 4-4): Suppose $M(s,k) = m$. Then there exists an $[m, m-s]$ code C with property B_k. Moreover, no column in a parity check matrix for C is entirely zero, otherwise $M(s,k) \leq m-1$. Puncture the code C in any position. Upon puncturing, a) above remains true trivially. It remains to show that we can find some $s-1$ cosets of the punctured code that generate $\mathbb{F}_2^{m-1}$, which is a direct result of Lemma 6. Hence the resulting $[m-1, m-s]$ code has property B_k, and it is a k-server PIR code. ■

TABLE I
LIST OF ALL 7 DIFFERENT TYPE OF COLUMNS USED IN CONSTRUCTING AN $(m,3)$-PIR CODE

h_a	h_b	h_c	h_x	h_y	h_z	h_w
$\begin{bmatrix}1\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\1\end{bmatrix}$	$\begin{bmatrix}0\\1\\1\end{bmatrix}$	$\begin{bmatrix}1\\0\\1\end{bmatrix}$	$\begin{bmatrix}1\\1\\0\end{bmatrix}$	$\begin{bmatrix}1\\1\\1\end{bmatrix}$

VI. ASYMPTOTIC BEHAVIOR OF CODED PIR

While deriving the precise values of $M(s,k)$ was our initial interest, studying the asymptotic behavior of $M(s,k)$ is no less interesting. In particular, lower bounds will help us to find constructions with the optimal storage overhead. Asymptotically, we will analyze the value of $M(s,k)$ when s is fixed and k is large, and vice versa. We briefly mention that we solved the first case while the lower bounds for the latter are yet to be found.

A. Storage Overhead for Fixed s

Let us first focus on the case where s, the ratio between the length of the whole data and the storage size of each server, is a fixed integer number, but k, the PIR protocol parameter, is large.

Theorem 9: For any pair of integer numbers s, and k, we have

$$M(s,k) \geq \frac{2^s-1}{2^{s-1}}k, \tag{31}$$

with equality if and only if k is divisible by 2^{s-1}.

Let us use the following example to illustrate the proof.

Example 8: Assume $s=3$, and C is an $(m,3)$ PIR code with k-server PIR property. The generator matrix of C contains m columns, each of length 3. The list of all possible options is shown in table I. Let us assume that the column multiplicities are given by $\mu_a, \mu_b, \mu_c, \mu_x, \mu_y, \mu_z$, and μ_w.

Since the code has the k-server PIR property, there should be k disjoint sets of columns each with h_a as their sum. h_a, h_b+h_z, h_c+h_y, and h_x+h_w are all such possibilities. It is easy to notice that there is no combination of the columns of type h_b, h_c, and h_x that would give h_a. So, each of the k sets should include at least one of the other columns. Therefore,

$$\mu_a+\mu_y+\mu_z+\mu_w \geq k.$$

Similar to $\{h_b, h_c, h_x\}$, we have three other sets $\{h_x, h_y, h_z\}$, $\{h_c, h_z, h_w\}$, and $\{h_b, h_y, h_w\}$ that are incapable of recovering the first data chunk by their own. So we have three more constraints

$$\mu_a+\mu_b+\mu_c+\mu_w \geq k,$$
$$\mu_a+\mu_b+\mu_x+\mu_y \geq k,$$
$$\mu_a+\mu_c+\mu_x+\mu_z \geq k.$$

Redoing the above argument for the second and the third information chunk, we get the following three *new* constraints

$$\mu_b+\mu_c+\mu_y+\mu_z \geq k,$$
$$\mu_b+\mu_x+\mu_z+\mu_w \geq k,$$
$$\mu_c+\mu_x+\mu_y+\mu_w \geq k;$$

TABLE II
COMPARISON OF THE CONSTRUCTIONS FOR $M(s,k)$ WITH RESPECT TO THE ASYMPTOTIC CODE REDUNDANCY $(M(s,k)-s)$

Code construction	Upper bound on $M(s,k)$	Asymptotic redundancy
Cubic construction	$M(s,k) \leqslant s+(k-1)\lceil s^{\frac{1}{k-1}}\rceil^{k-2}$	$O(s^{1-\frac{1}{k-1}})$
Steiner System	$M\left(\frac{n(n-1)}{(k-1)(k-2)},k\right) \leqslant n+\frac{n(n-1)}{(k-1)(k-2)}$	$O(s^{\frac{1}{2}})$
Type-1 DTI codes (1)	$M\left(2^{2\theta\ell}-(2^{\theta+1}-1)^{\ell},2^{\ell}+2\right) \leqslant 2^{2\theta\ell}-1$	$O(s^{\frac{1}{2}})$
Type-1 DTI codes (2)	$M\left((2^{\lambda}-1)^{\ell}-1,2^{\ell}\right) \leqslant 2^{\lambda\ell}-1$	$O(s^{1-\frac{1}{\ell}})$
Constant weight codes	$M\left(\binom{n}{2},3\right) \leqslant \binom{n}{2}+n$	$O(s^{\frac{1}{2}})$

And, by adding all the above constraints we have

$$M(3,k)=m=\mu_a+\mu_b+\mu_c+\mu_x+\mu_y+\mu_z+\mu_w \geq \frac{7}{4}k.$$

It is trivial that when k is divisible by 4, setting $\mu_a=\mu_b=\mu_c=\mu_x=\mu_y=\mu_z=\mu_w=\frac{k}{4}$ gives the equality. We can indeed use the results from Lemmas 4, and 5 to prove that $M(3,k)=\lceil\frac{7k}{4}\rceil$.

Proof of Theorem 9: For the general s, the generator matrix contains at most 2^s-1 different non-zero columns. Assume C is a k-server PIR code with length m and dimension s. Therefore, for each $1\leq i\leq s$, one can find k disjoint subsets of the columns with their equal to

$$e_i=\left(\underbrace{0,\ldots,0}_{i-1},1,0,\ldots,0\right)^t.$$

Similar to the example, we look at all the $(s-1)$-dimensional subspaces V in $\mathbb{F}_2^s$, such that $e_i\notin V$. It is clear that no combination of the columns in V can retrieve e_i. So, each of the k subsets should include at least one vector from V^c, where V^c denotes the complement of V in $\mathbb{F}_2^s$. Now let V be a subspace of $\mathbb{F}_2^s$ that does not contain the unit vector e_i. Then $\sum_{v\in V^c}\mu_v\geq k$ is a constraint involving μ_{e_i}.

There are

$$\frac{(2^s-2)(2^s-4)\cdots\left(2^s-2^{s-2}\right)}{\left(2^{s-1}-1\right)\left(2^{s-1}-2\right)\cdots\left(2^{s-1}-2^{s-2}\right)}=2^{s-1}$$

such subspaces for each i, which gives us 2^{s-1} constraints for each e_i. It suffices to show that there are exactly 2^s-1 unique constraints after merging all these sets. Now we recall that the non-zero codewords of the simplex code of length 2^s-1 are precisely the supports of all sets of the form V^c, where V is an $(s-1)$-dimensional subspace of $\mathbb{F}_2^s$ (see [13, p. 380] for proof.) It is now clear that we have 2^s-1 unique constraints since there are exactly 2^s-1 codewords of weight 2^{s-1} in the simplex code of length 2^s-1. Moreover, exactly 2^{s-1} of these codewords have value 1 in a fixed coordinate. In other words, we get the vector $2^{s-1}\mathbf{1}_{2^s-1}$ as the sum of all these codewords. So,

$$2^{s-1}\sum_{v\in\mathbb{F}_2^s}\mu_v\geq\left(2^s-1\right)k \iff$$

$$M(s,k)=m=\sum_{v\in\mathbb{F}_2^s}\mu_v\geq\frac{(2^s-1)}{2^{s-1}}k. \qquad \blacksquare$$

Let us fix s. The introduced lower bound in (31) along with its equality condition shows that $M(s,k)=O(k)$, when k becomes large. In other words $M(s,k)\sim 2k$ for small s and k large.

B. Storage Overhead for Fixed k

It was already shown that for fixed k, there are elementary constructions to achieve storage overhead $\frac{M(s,k)}{s}$ arbitrary close to 1. We are yet to determine how fast it decreases. Table II summarizes the constructions introduced in the previous section with their asymptotic behavior. Note that, the explicit formula for the constructions based on constant-weight codes is only known for $k=3$.

We observe that non of the introduced constructions achieves storage overhead less than $1+O(s^{-\frac{1}{2}})$. However, it is not clear if that is the optimum value one can get. So far, the best (and trivial) lower bound is given by

$$M(s,k)\geq s+O(\log s).$$

VII. PIR ARRAY CODES

In all the constructions we presented so far, we assumed that the database was partitioned into s parts, where every server stores n/s bits that were considered to be a single symbol. In this section we seek to extend this idea and let every server store more than a single symbol. For example, we can partition the database into $2s$ parts of $n/(2s)$ bits each such that every server stores two symbols. This can be generalized such that every server stores a fixed number of symbols. One of the benefits of this method to construct PIR codes is that we can support setups in which the number of bits stored in a server is n/s where s is not necessarily an integer. Furthermore, we will show that it is also possible to improve, for some instances of s and k, the value of $M(s,k)$ and hence the storage overhead as well. Since every server stores more than a single symbol we treat the code construction as an *array code* and thus we call these codes *PIR array codes*. When a server receives a query q then it responses with multiple answers corresponding to the number of symbols stored in the server. We illustrate the idea of PIR array codes in the next example.

Example 9: Assume that the database $\boldsymbol{x}$ is partitioned into 12 parts $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_{12}$ which are stored in four servers as follows:

*Server*1	*Server*2	*Server*3	*Server*4
$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_1+\boldsymbol{x}_2+\boldsymbol{x}_3$
$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_1$	$\boldsymbol{x}_6$
$\boldsymbol{x}_4$	$\boldsymbol{x}_5$	$\boldsymbol{x}_4+\boldsymbol{x}_5+\boldsymbol{x}_6$	$\boldsymbol{x}_4$
$\boldsymbol{x}_5$	$\boldsymbol{x}_6$	$\boldsymbol{x}_8$	$\boldsymbol{x}_9$
$\boldsymbol{x}_7$	$\boldsymbol{x}_7+\boldsymbol{x}_8+\boldsymbol{x}_9$	$\boldsymbol{x}_9$	$\boldsymbol{x}_7$
$\boldsymbol{x}_8$	$\boldsymbol{x}_{11}$	$\boldsymbol{x}_{11}$	$\boldsymbol{x}_{12}$
$\boldsymbol{x}_{10}+\boldsymbol{x}_{11}+\boldsymbol{x}_{12}$	$\boldsymbol{x}_{11}$	$\boldsymbol{x}_{12}$	$\boldsymbol{x}_{10}$

Thus, every server stores 7 parts, each of $n/12$ bits, so $\frac{n}{12/7}$ bits are stored in each server and the storage overhead is $7/3$. Using this code, it is possible to invoke a 3-server linear protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$. Assume Alice seeks to read the bit $x_{1,i}$ for $i \in [n/12]$, she invokes the algorithm $\mathcal{Q}$ to receive three queries $\mathcal{Q}(3, n/12; i) = (q_1, q_2, q_3)$. The first sever is assigned with the query q_1, the second and fourth servers with the query q_2 and the third server with the query q_3. Each server responds with 7 answers corresponding to the 7 parts it stores. Alice receives all 28 answers but only needs 5 answers to retrieve the value of $x_{1,i}$. From the first server she receives the answer $a_1 = \mathcal{A}(3, 1, \boldsymbol{x}_1, q_1)$, from the second server she receives two answers $a_2' = \mathcal{A}(3, 2, \boldsymbol{x}_2, q_2)$ and $a_2'' = \mathcal{A}(3, 2, \boldsymbol{x}_3, q_2)$, from the third server $a_3 = \mathcal{A}(3, 3, \boldsymbol{x}_3, q_3)$, and lastly from the fourth server she receives $a_4 = \mathcal{A}(3, 2, \boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3, q_2)$. Note that from the linearity of the protocol $\mathcal{P}$, we have

$$\begin{aligned} &a_2' + a_2'' + a_4 \\ &\quad = \mathcal{A}(3, 2, \boldsymbol{x}_2, q_2) + \mathcal{A}(3, 2, \boldsymbol{x}_3, q_2) \\ &\qquad + \ \mathcal{A}(3, 2, \boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3, q_2) \\ &\quad = \mathcal{A}(3, 2, \boldsymbol{x}_1, q_2), \end{aligned}$$

and thus $x_{1,i}$ is retrieved by applying the algorithm $\mathcal{C}$

$$x_{1,i} = \mathcal{C}\big(3, n/12; i, a_1, a_2' + a_2'' + a_4, a_3\big).$$

In the last example, we see that we repeated the same code four times. That was done in order to guarantee that the number of symbols stored in each server is the same. We could instead show only the first two rows of the first code and then claim that by interleaving of the column which stores only one symbol it is possible to guarantee that each server stores the same number of symbols. While we saw that in this example it is possible to construct PIR codes with more flexible parameters, the download communication was increased and we needed only 5 out of the 28 received answers. However, since the number of symbols in each server is fixed (and will be in the constructions in this section) the communication complexity order is not changed.

In general, we refer to an $m_1 \times m_2$ array code as a scheme to encode s information bits $x_1, \ldots, x_s$ into an array of size $m_1 \times m_2$. An $(m_1 \times m_2, s)$-server coded PIR protocol is defined in a similar way to Definition 3. We formally define PIR array codes.

Definition 7: A binary $[m_1 \times m_2, s]$ linear code will be called a ***k*-server PIR array code** if for every information bit x_i, $i \in [s]$, there exist k mutually disjoint sets $R_{i,1}, \ldots, R_{i,k} \subseteq [m_2]$ such that for all $j \in [k]$, x_i is a linear function of the bits stored in the columns of the set $R_{i,j}$.

Very similarly to Theorem 1 we conclude that if there exists an $[m_1 \times m_2, s]$ k-server PIR array code and a k-server linear PIR protocol $\mathcal{P}$ then there exists an $(m_1 \times m_2, s)$-server coded PIR protocol that can emulate the protocol $\mathcal{P}$. Next, we give another example of PIR array code which explicitly improves the storage overhead.

Example 10: We give here a construction of $[2 \times 25, 6]$ 15-server PIR array code. The 6 information bits are denoted by $x_1, x_2, x_3, x_4, x_5, x_6$ and are stored in a 2×25 array as follows:

The first row specifies the server number. The other two rows indicate the bits which are stored in each column. It is possible to verify that this construction provides a 15-server PIR array code. For example for the first bit we get the following 15 sets:

$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6, 16\}, \{7, 17\}, \{8, 18\}, \{9, 19\},$$
$$\{10, 20\}, \{11, 21\}, \{12, 22\}, \{13, 23\}, \{14, 24\}, \{15, 25\}$$

where it is possible to retrieve the value of x_1 by the bits stored in the columns of each group.

The number of bits stored in each server of this example is $n/3$ and thus $s = 3$. If we had to use the best construction of an $[m, 3]$ 15-server PIR code, then $M(3, 15) = 26$ servers are required, while here we used only 25 servers. Hence, we managed to improve the storage overhead for $s = 3$ and $k = 15$.

We extend Example 10 to a general construction of PIR array code. Let t be a fixed integer $t \geq 2$. The number of information bits is $s = t(t+1)$, the number of rows is $m_1 = t$ and the number of columns is $m_2 = m_2' + m_2''$, where $m_2' = \binom{t(t+1)}{t}$ and $m_2'' = \binom{t(t+1)}{t+1}/t$. In the first m_2' columns we simply store all tuples of t bits out of the $t(t+1)$ information bits. In the last m_2'' columns we store all possible summations of $t+1$ bits. There are $\binom{t(t+1)}{t+1}$ such summations and since there are t rows, t summations are stored in each column, so the number of columns for this part is $m_2'' = \binom{t(t+1)}{t+1}/t$. We also require that in the last m_2'' columns every bit appears in exactly one summation. Note that Example 10 is a special case of this construction for $t = 2$. A code generated by this construction will be denoted by $\mathcal{C}_{A-PIR}(t)$. The proof of this theorem appears in [10, Th. 10].

Theorem 10: For any integer $t \geq 2$, the code $\mathcal{C}_{A-PIR}(t)$ is an $[m_1 \times m_2, t(t+1)]$ k-server PIR array code where

$$s = t + 1, m_1 = t,$$
$$m_2 = \binom{t(t+1)}{t} + \frac{\binom{t(t+1)}{t+1}}{t}, k = \binom{t(t+1)}{t},$$

and its storage overhead is

$$\frac{\binom{t(t+1)}{t} + \binom{t(t+1)}{t+1}/t}{t+1}.$$

Table IV compares the improvement in the number of servers, and thus storage overhead, when using the PIR array code $\mathcal{C}_{A-PIR}(t)$. For $t = 2$ and $s = 3, k = 15$ we know the exact value of $M(s, k)$ calculated above, and for all other values of t we get a lower bound on the value of $M(s, k)$ according to Theorem 9.

In this section we demonstrated how PIR array codes are beneficial for the purpose of improving the storage overhead. Another possible direction for improvement is to explore how PIR array codes can be used to reduce the communication complexity while fixing the storage overhead. We believe that this is possible since based on Table IV we can expect that if the number of servers remains the same, then the value of k can be reduced and this will imply an improvement in the communication complexity. However, we leave this problem for future research.

TABLE III
A CONSTRUCTION OF $[2 \times 25, 6]$ 15-SERVER PIR ARRAY CODE

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	x_1	x_1	x_1	x_1	x_2	x_2	x_2	x_2	x_3	x_3	x_3	x_4	x_4	x_5
x_2	x_3	x_4	x_5	x_6	x_3	x_4	x_5	x_6	x_4	x_5	x_6	x_5	x_6	x_6

16	17	18	19	20	21	22	23	24	25
$x_1+x_2+x_3$	$x_1+x_2+x_4$	$x_1+x_2+x_5$	$x_1+x_2+x_6$	$x_1+x_3+x_4$	$x_1+x_3+x_5$	$x_1+x_3+x_6$	$x_1+x_4+x_5$	$x_1+x_4+x_6$	$x_1+x_5+x_6$
$x_3+x_4+x_5$	$x_3+x_5+x_6$	$x_3+x_4+x_6$	$x_3+x_4+x_5$	$x_2+x_5+x_6$	$x_2+x_4+x_6$	$x_2+x_4+x_5$	$x_2+x_3+x_6$	$x_2+x_3+x_5$	$x_2+x_3+x_4$

TABLE IV
COMPARISON BETWEEN THE CODE $\mathcal{C}_{A-PIR}(t)$ AND THE CORRESPONDING BEST VALUES OF $M(s,k)$

t	s	k	m_2	$M(s,k)$
2	3	15	25	26
3	4	220	385	$\geqslant 413$
4	5	4845	8721	$\geqslant 9387$
5	6	142506	261261	$\geqslant 280559$

VIII. CONCLUSION AND DISCUSSION

A new framework to utilize private information retrieval in distributed storage systems is introduced in this paper. The new scheme is based on the idea of using coding instead of the replications in the traditional PIR protocols, when the storage size of each server is much less than the size of the database. We have shown that among the three main parameters in measuring the quality of k-server PIR protocols, i.e., *communication complexity, computation complexity,* and *storage overhead*, the first two remain the same and the latter improves significantly in the asymptotic regime. In particular, for a fixed k and a limited server size, the storage overhead becomes $1+o(1)$ as the number of servers becomes large. The optimal storage overhead with the coded PIR is also studied and the explicit value is derived for many cases. The presented constructions lead to coded PIR schemes with storage overhead $1+O(s^{-1/2})$ for any fixed k, where s is the ratio between the size of the database and the storage size of each server. This result has been established to be optimal in [22], [32], [45] and several more results in terms of bounds and code constructions have been rigorously studied since the first publication of this paper 2015; see, e.g., [1], [17], [27], [34], [40].

Another research direction that has been explored is the construction of PIR array codes such as the ones given in Section VII. These constructions are examples for improvements either in the storage overhead or the existence of codes with other parameters which cannot be achieved by the non-array PIR codes. Several more code constructions have been proposed in the literature in the past year; see, e.g., [5], [10], [28], [48], and yet we hope that more constructions will appear to further improve these parameters.

Lastly, we note that the family of PIR codes that has been studied in the paper is different than the line of works that studied the PIR capacity and the PIR codes for this model. The PIR codes that are referred to in the paper are motivated by PIR protocols that optimize both the upload and download complexity. Thus, these codes are targeted to solve a completely different problem that the ones that address the PIR capacity and optimize *only* the download complexity; see [4], [9], [24], [26], [35], [36], [37], [39] and references therein. Furthermore, PIR codes are similar in their definition to locally repairable codes with availability [29], [30], [42], with the important distinction that PIR codes do not impose any constraint on the size of the recovering sets as done for LRCs. Hence, it is not possible to compare between PIR codes and the above families of codes.

ACKNOWLEDGMENT

The authors thank Anna Gál, Amos Beimel, Han Mao Kiah, Eyal Kushilevitz, Sankeerth Rao, Itzhak Tamo, and Mary Wootters for very helpful discussions. They also thank the reviewers and editors for their insightful comments that contributed to the improvement of the quality of this work.

REFERENCES

[1] H. Asi and E. Yaakobi, "Nearly optimal constructions of PIR and batch codes," *IEEE Trans. Inf. Theory*, vol. 65, no. 2, pp. 947–964, Feb. 2019.
[2] A. Ambainis, "Upper bound on the communication complexity of private information retrieval," in *Proc. Intern. Colloq. Automata, Languages Program.*, Graz, Austria, July 1997, pp. 401–407.
[3] D. Augot, F. Levy-dit-Vehel, and A. Shikfa, "A storage-efficient and robust private information retrieval scheme allowing few servers," in *Proc. 13th Intern. Conf. Cryptol. Netw. Secur.*, Heraklion, Greece, Oct. 2014, pp. 222–239.
[4] K. Banawan and S. Ulukus, "The capacity of private information retrieval from coded databases," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1945–1956, Mar. 2018.
[5] S. R. Blackburn and T. Etzion, "PIR array codes with optimal virtual server rate," *IEEE Trans. Inf. Theory*, vol. 65, no. 10, pp. 6136–6145, Oct. 2019.
[6] A. E. Brouwer. "Bounds for Binary Constant Weight Codes." [Online]. Available: http://www.win.tue.nl/∼aeb/codes/Andw.html
[7] A. Beimel, Y. Ishai, and E. Kushilevitz, "General constructions for information-theoretic private information retrieval," *J. Comput. Syst. Sci.*, vol. 71, no. 2, pp. 213–247, 2005.
[8] A. Beimel, Y. Ishai, E. Kushilevitz, and J.-F. Raymond, "Breaking the $O(n^{1/(2k-1)})$ barrier for information theoretic private information retrieval," in *Proc. 43rd IEEE Symp. Foundations Comput. Sci.*, Vancouver, Canada, Nov. 2002, pp. 261–270.
[9] T. H. Chan, S.-W. Ho, and H. Yamamoto, "Private information retrieval for coded storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2015, pp. 2842–2846.
[10] Y. M. Chee, H. Mao Kiah, E. Yaakobi, and H. Zhang, "A generalization of Blackburn-Etzion construction for private information retrieval array codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Paris, France, Jul. 2019, pp. 1062–1066.
[11] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. 36th IEEE Symp. Foundations Comput. Sci.*, Milwaukee, WI, USA, Oct. 1995, pp. 41–50.
[12] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, 1998.
[13] D. J. Costello and S. Lin, *Error Control Coding*, 2nd ed., London, U.K.: Pearson, 2004.

[14] Z. Dvir and S. Gopi, "2-server PIR with sub-polynomial communication," in *Proc. 47th Annu. ACM Symp. Theory Comput. (STOC'15)*, New York, NY, USA, 2015, pp. 577–584.
[15] K. Efremenko, "3-query locally decodable codes of subexponential length," in *Proc. 41st ACM Symp. Theory Comput.*, Bethesda, MD, USA, Jun. 2009, pp. 39–44.
[16] A. Fazeli, A. Vardy, and E. Yaakobi, "Codes for distributed PIR with low storage overhead," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2015, pp. 2852–2856.
[17] S. L. Frank-Fischer, V. Guruswami, and M. Wootters, "Locality via partially lifted codes," 2017, *arXiv:1704.08627*.
[18] W. Gasarch, "A survey on private information retrieval," *Bull. Eur. Assoc. Theoret. Comput. Sci.*, vol. 82, nos. 72–107, p. 113, 2004.
[19] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Nov. 2012.
[20] P. Huang, E. Yaakobi, H. Uchikawa, and P. H. Siegel, "Linear locally repairable codes with availability," *IEEE Trans. Inf. Theory*, vol. 62, no. 11, pp. 6268–6283, Nov. 2016.
[21] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," in *Proc. 36th ACM Symp. Theory Comput.*, Chicago, IL, USA, Jun. 2004, pp. 262–271.
[22] S. R. Karingula, A. Vardy, and M. Wootters, "Lower bounds on the redundancy of linear codes with disjoint repair groups," in *Proc. IEEE Symp. Inf. Theory*, Espoo, Finland, Jun. 2022, pp. 975–979.
[23] S. Kopparty, S. Saraf, and S. Yekhanin, "High-rate codes with sublinear-time decoding," in *Proc. 43rd ACM Symp. Theory Comput.*, San Jose, CA, USA, Jun. 2011, pp. 167–176.
[24] S. Kumar, H.-Y. Lin, E. Rosnes, and A. Graell i Amat, "Achieving maximum distance separable private information retrieval capacity with linear codes," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4243–4273, Jul. 2019.
[25] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: Single database, computationally-private information retrieval," in *Proc. 38th IEEE Symp. Foundations Comput. Sci.*, Miami Beach, FL, USA, Oct. 1997, pp. 364–373.
[26] H.-Y. Lin, S. Kumar, E. Rosnes, A. Graell i Amat, and E. Yaakobi, "Weak private information retrieval," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 1257–1261.
[27] H.-Y. Lin and E. Rosnes, "Lengthening and extending binary private information retrieval codes," in *Proc. Int. Zurich Seminar Inf. Commun.*, Zurich, Switzerland, Feb. 2018, pp. 113–117.
[28] M. Nassar and E. Yaakobi, "Array codes for functional PIR and batch codes," *IEEE Trans. Inf. Theory*, vol. 68, no. 2, pp. 839–862, Feb. 2022.
[29] L. Pamies-Juarez, H. D. L. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," in *Proc. IEEE Symp. Inf. Theory*, Istanbul, Turkey, Jul. 2013, pp. 892–896.
[30] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, and S. Vishwanath, "Locality and availability in distributed storage," *IEEE Trans. Inf. Theory*, vol. 62, no. 8, pp. 4481–4493, Aug. 2016.
[31] A. S. Rawat, Z. Song, A. G. Dimakis, and A. Gál, "Batch codes through dense graphs without short cycles," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1592–1604, Apr. 2016.
[32] S. Rao and A. Vardy, "Lower bound on the redundancy of PIR codes," 2016, *arXiv:1605.01869*.
[33] N. B. Shah, K. V. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," in *Proc. IEEE Symp. Inf. Theory*, Honolulu, HI, USA, Jul. 2014, pp. 856–890.
[34] V. Skachek, "Batch and PIR codes and their connections to locally repairable codes," in *Network Coding and Subspace Designs*. Cham, Switzerland: Springer, 2018, pp. 427–442.
[35] H. Sun and S. A. Jafar, "The capacity of private information retrieval," *IEEE Trans. Inf. Theory*, vol. 63, no. 7, pp. 4075–4088, Jul. 2017.
[36] H. Sun and S. A. Jafar, "The capacity of robust private information retrieval with colluding databases," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2361–2370, Apr. 2018.
[37] R. Tajeddine, O. W. Gnilke, and S. El Rouayheb, "Private information retrieval from MDS coded data in distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 64, no. 11, pp. 7081–7093, Nov. 2018.
[38] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4661–4676, Aug. 2014.
[39] C. Tian, H. Sun, and J. Chen, "Capacity-achieving private information retrieval codes with optimal message size and upload cost," *IEEE Trans. Inf. Theory*, vol. 65, no. 11, pp. 7613–7627, Nov. 2019.
[40] M. Vajha, V. Ramkumar, and P. Vijay Kumar, "Binary, shortened projective reed muller codes for coded private information retrieval," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2017, pp. 2648–2652.
[41] A. Vardy and E. Yaakobi, "Constructions of batch codes with near-optimal redundancy," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 1197–1201.
[42] A. Wang, Z. Zhang, and M. Liu, "Achieving arbitrary locality and availability in binary codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, China, Jun. 2015, pp. 1866–1870.
[43] R. M. Wilson, "An existence theory for pairwise balanced designs," *J. Comb. Theory Ser. A*, vol. 13, pp. 220–273, Sep. 1972.
[44] D. P. Woodruff and S. Yekhanin, "A geometric approach to information-theoretic private information retrieval," *SIAM J. Comput.*, vol. 37, no. 4, pp. 1046–1056, 2007.
[45] M. Wootters, "Linear codes with disjoint repair groups," unpublished.
[46] S. Yekhanin, "Private information retrieval," *Commun. ACM*, vol. 53, no. 4, pp. 68–73, 2010.
[47] S. Yekhanin, "Towards 3-query locally decodable codes of subexponential length," *J. ACM*, vol. 55, no. 1, pp. 1–16, 2008.
[48] Y. Zhang, X. Wang, H. Wei, and G. Ge, "On private information retrieval array codes," *IEEE Trans. Inf. Theory*, vol. 65, no. 9, pp. 5565–5573, Sep. 2019.

Alexander Vardy was born in Moscow, Russia, in 1963. He received the B.Sc. degree (summa cum laude) from Technion, Israel, in 1985, and the Ph.D. degree from Tel-Aviv University, Israel, in 1991.

From 1985 to 1990, he was with Israeli Air Force, where he worked on electronic counter measures systems and algorithms. From 1992 to 1993, he was a Visiting Scientist with IBM Almaden Research Center, San Jose, CA, USA. From 1993 to 1998, he was with the University of Illinois at Urbana–Champaign first as an Assistant Professor and then as an Associate Professor. Since 1998, he has been with the University of California at San Diego (UCSD), where he is currently the Jack Keil Wolf Chair Professor with the Department of Electrical and Computer Engineering and the Department of Computer Science. While on sabbatical from UCSD, he has held long-term visiting appointments with CNRS, France, EPFL, Switzerland, the Technion—Israel Institute of Technology, and Nanyang Technological University, Singapore. His research interests include error-correcting codes, algebraic and iterative decoding algorithms, lattices and sphere packings, coding for storage systems, cryptography, computational complexity theory, and fun math problems. He received the IBM Invention Achievement Award in 1993 and the NSF Research Initiation and CAREER Awards in 1994 and 1995, respectively. In 1996, he was appointed as a Fellow of the Center for Advanced Study, University of Illinois, and received the Xerox Award for Faculty Research. He received the IEEE Information Theory Society Paper Award (jointly with Ralf Koetter) in 2004. In 2005, he received the Fulbright Senior Scholar Fellowship and the Best Paper Award at the IEEE Symposium on Foundations of Computer Science. In 2017, his work on polar codes was recognized by the IEEE Communications and Information Theory Societies Joint Paper Award. From 1995 to 1998, he was an Associate Editor of *Coding Theory*. From 1998 to 2001, he was an Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION THEORY. He has been a member of the Board of Governors of the IEEE Information Theory Society from 1998 to 2006 and from 2011 to 2017. In 1996, he became a Fellow of the David and Lucile Packard Foundation.

Eitan Yaakobi (Senior Member, IEEE) received the B.A. degree in computer science and mathematics and the M.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California at San Diego, San Diego, in 2011. He is an Associate Professor with the Computer Science Department, Technion—Israel Institute of Technology, where he also holds a courtesy appointment with Electrical and Computer Engineering Department. From 2011 to 2013, he was a Postdoctoral Researcher with the Department of Electrical Engineering, California Institute of Technology and the Center for Memory and Recording Research, University of California at San Diego. His research interests include information and coding theory with applications to nonvolatile memories, associative memories, DNA storage, data storage and retrieval, and private information retrieval. He is a recipient of several grants, including the ERC Consolidator Grant. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship in 2010 and 2011. Since 2016, he has been affiliated with the Center for Memory and Recording Research, University of California at San Diego and since 2018, he has been affiliated with the Institute of Advanced Studies, Technical University of Munich, where he holds a four-year Hans Fischer Fellowship, funded by the German Excellence Initiative and the EU 7th Framework Program. Since 2020, he has been serving as an Associate Editor for Coding and Decoding of the IEEE TRANSACTIONS ON INFORMATION THEORY.