Array Codes for Functional PIR and Batch Codes

Mohammad Nassar, *Student Member, IEEE*, and Eitan Yaakobi, *Senior Member, IEEE*

***Abstract*— A *functional PIR array code* is a coding scheme which encodes some s information bits into a $t \times m$ array such that every linear combination of the s information bits has k mutually disjoint *recovering sets*. Every recovering set consists of some of the array's columns while it is allowed to read at most ℓ encoded bits from every column in order to receive the requested linear combination of the information bits. *Functional batch array codes* impose a stronger property where every multiset request of k linear combinations has k mutually disjoint recovering sets. *Locality functional array codes* demand that the size of every recovering set is restrained to be at most r. Given the values of s, k, t, ℓ, r, the goal of this paper is to study the optimal value of the number of columns m such that these codes exist. Several lower bounds are presented as well as explicit constructions for several of these parameters.**

***Index Terms*— Private information retrieval (PIR) codes, batch codes, codes with availability, covering codes.**

I. Introduction

PRIVATE information retrieval (PIR) codes and *batch codes* are families of codes which have several applications such as PIR protocols [3], [10], [15], [19], [26], [42], [45], [48], erasure codes in distributed storage systems [30], [31], [38], one-step majority-logic decoding [24], [28], load balancing in storage, cryptographic protocols [22], switch codes [6], [11], [43], and more. They have been recently generalized to *functional PIR* and *functional batch* codes [51]. In this work we study these families of codes when they are used as array codes.

The setup of storing information in array codes works as follows. Assume s bits are encoded to be stored in a $t \times m$ array, where each column corresponds to a *server* that stores the encoded bits. The encoded bits should satisfy several properties which depend upon whether the resulting code is a PIR, batch, functional PIR, or functional batch codes. Given a design parameter k of the code, it is required in PIR codes that every information bit has k mutually disjoint *recovering sets*. Here, a recovering set is a set of columns, i.e., servers, in which given the encoded bits in the columns of the recovering set it is possible to recover the information bit. In case it is possible to read only a portion of the encoded bits in every column, we denote this parameter by ℓ. An array code with these parameters and properties is defined as an (s, k, m, t, ℓ) *PIR array code*. Furthermore, it will be called an (s, k, m, t, ℓ) *batch array code* if every *multiset* request of k information bits has k mutually disjoint recovering sets. In case the requests are not only of information bits but any linear combination of them, we receive an (s, k, m, t, ℓ) *functional PIR array code*, if every linear combination has k mutually disjoint recovering sets or (s, k, m, t, ℓ) *functional batch array code* for a multiset request of k linear combinations. Yet another family of codes that will be studied in this paper will be referred by *locality functional array codes*. Here we assume that $\ell = t$ and an (s, k, m, t, r) *locality functional array code* guarantees that every linear combination $\boldsymbol{v}$ of the information bits has k mutually disjoint recovering sets, where each is of size of at most r.

The main figure of merit when studying these families of codes is to optimize the number of columns, i.e., servers, given the values of s, k, t, ℓ. Thus, the smallest m such that an (s, k, m, t, ℓ) PIR, batch, functional PIR, functional batch code exists, is denoted by $P_{t,\ell}(s, k), B_{t,\ell}(s, k), FP_{t,\ell}(s, k), FB_{t,\ell}(s, k)$, respectively. Studying the value of $P_{t,\ell}(s, k)$ has been initiated in [19] and since then several more results have appeared; see e.g. [4], [5], [9], [50]. Note that the first work [22] which studied batch codes defined them in their array codes setup and only later on they were studied in their one-dimensional case, also known as *primitive batch codes*; see e.g. [1], [27], [32], [41], [49]. Functional PIR and batch codes have been recently studied in [51] but only for vectors, that is, $t = \ell = 1$. Thus, this paper initiates the study of functional PIR and batch codes in the array setup.

The motivation to study functional PIR and batch codes originates from the observation that in many cases and protocols, such as PIR, the user is not necessarily interested in one of the information bits, but rather, some linear combination of them. Furthermore, functional batch codes are closely related to the family of *random I/O (RIO) codes*, introduced by Sharon and Alrod [33], which are used to improve the random input/output performance of flash memories. A variant of RIO codes, called *parallel RIO codes*, was introduced in [46], and linear codes of this family of codes have been studied in [47]. It was then shown in [51] that in fact linear parallel RIO codes are equivalent to functional batch codes.

The rest of the paper is organized as follows. In Section II, we formally define the codes studied in the paper, discuss

Manuscript received September 8, 2020; revised August 17, 2021; accepted October 20, 2021. Date of publication November 2, 2021; date of current version January 20, 2022. 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. An earlier version of this paper was presented in part at the 2020 IEEE International Symposium on Information Theory (ISIT) (reference [29]) [DOI: 10.1109/ISITaaa84.2020.9174459]. *(Corresponding author: Eitan Yaakobi.)*

The authors are with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 3200003, Israel (e-mail: mohamadtn@cs.technion.ac.il; yaakobi@cs.technion.ac.il).

Communicated by C. Hollanti, Associate Editor for Coding Theory.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2021.3124925.

Digital Object Identifier 10.1109/TIT.2021.3124925

some of the previous related work, and list several basic properties. In Section IV, we show lower bounds on the number of servers for functional PIR and batch array codes. Section V lists several code constructions which are based on the Gadget Lemma, covering codes, and several more results for $k = 1, 2$. Section VI presents three constructions of array codes and in Section VII the rates of these codes are studied. Section VIII studies locality functional array codes. Lastly, Section IX concludes the paper.

II. Definitions and Preliminaries

This work is focused on five families of codes, namely *private information retrieval* (*PIR*) codes that were defined recently in [19], *batch codes* that were first studied by Ishai *et al.* in [22], their extension to *functional PIR codes* and *functional batch codes* that was investigated in [51], *and locality functional codes*. In these five families of codes, s information bits are encoded to m bits. While for PIR codes it is required that every information bit has k mutually disjoint recovering sets, batch codes impose this property for every multiset request of k bits. Similarly, for functional PIR codes it is required that every linear combination of the information bits has k mutually disjoint recovering sets, and functional batch codes impose this property for every multiset request of k linear combination of the bits. Lastly, similar to functional PIR codes, for locality functional codes it is required that the size of every recovering set is limited to be at most r. While this description of the codes corresponds to the case of one-dimensional codewords, the goal of this work is to study their extension as *array codes*, which is defined as follows. The set $[n]$ denotes the set of integers $\{1, 2, \ldots, n\}$ and $\Sigma = \mathbb{F}_2$.

We start with the formal definition of the first four families of codes that will be studied in the paper, while we defer the definition of locality functional array codes to Section VIII.

Definition 1:

a) An (s, k, m, t, ℓ) **PIR array code** over Σ is defined by an encoding map $\mathcal{E} : \Sigma^s \to (\Sigma^t)^m$ that encodes s information bits $x_1, \ldots, x_s$ into a $t \times m$ array and a decoding function $\mathcal{D}$ that satisfies the following property. For any $i \in [s]$ there is a partition of the columns into k recovering sets $S_1, \ldots, S_k \subseteq [m]$ such that x_i can be recovered by reading at most ℓ bits from each column in $S_j, j \in [k]$.

b) An (s, k, m, t, ℓ) **batch array code** over Σ is defined by an encoding map $\mathcal{E} : \Sigma^s \to (\Sigma^t)^m$ that encodes s information bits $x_1, \ldots, x_s$ into a $t \times m$ array and a decoding function $\mathcal{D}$ that satisfies the following property. For any multiset request of k bits $i_1, \ldots, i_k \in [s]$ there is a partition of the columns into k recovering sets $S_1, \ldots, S_k \subseteq [m]$ such that $x_{i_j}, j \in [k]$ can be recovered by reading at most ℓ bits from each column in S_j.

c) An (s, k, m, t, ℓ) **functional PIR array code** over Σ is defined by an encoding map $\mathcal{E} : \Sigma^s \to (\Sigma^t)^m$ that encodes s information bits $x_1, \ldots, x_s$ into a $t \times m$ array and a decoding function $\mathcal{D}$ that satisfies the following property. For any request of a linear combination $\boldsymbol{v}$ of the information bits, there is a partition of the columns into k recovering sets $S_1, \ldots, S_k \subseteq [m]$ such that $\boldsymbol{v}$ can be recovered by reading at most ℓ bits from each column in $S_j, j \in [k]$.

d) An (s, k, m, t, ℓ) **functional batch array code** over Σ is defined by an encoding map $\mathcal{E} : \Sigma^s \to (\Sigma^t)^m$ that encodes s information bits $x_1, \ldots, x_s$ into a $t \times m$ array and a decoding function $\mathcal{D}$ that satisfies the following property. For any multiset request of k linear combinations $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ of the information bits, there is a partition of the columns into k recovering sets $S_1, \ldots, S_k \subseteq [m]$ such that $\boldsymbol{v}_j, j \in [k]$ can be recovered by reading at most ℓ bits from each column in S_j.

We refer to each column as a *bucket* and to each entry in a bucket as a *cell*. Furthermore, it is said that a cell stores a *singleton* if one of the information bits is stored in the cell. In the rest of the paper we will refer to every linear combination of the information bits as a binary vector of length s, which indicates the information bits in this linear combination. Our goal is to fix the values of s, k, t and ℓ and then seek to optimize the value of m. In particular, we will have that t and ℓ are fixed, where $t \geqslant \ell$, and then study the growth of m as a function of s and k. Hence, we denote by $P_{t,\ell}(s, k), B_{t,\ell}(s, k), FP_{t,\ell}(s, k), FB_{t,\ell}(s, k)$ the smallest m such that an (s, k, m, t, ℓ) PIR, batch, functional PIR, functional batch code exists, respectively. In case $\ell = t = 1$ we will simply remove them from these notations.

Remark 1: The family of PIR codes that is 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 the family of codes that was first studied in [19] for 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 [2], [8], [23], [25], [35]–[37], [39] and references therein. Furthermore, PIR codes are similar in their definition to locally repairable codes with availability [30], [31], [44], 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.

III. Previous Work and Basic Results

The following upper and lower bounds on the number of buckets for PIR array codes have been shown in [5], [9], [50] and are stated in the following theorem.

Theorem 2:

a) $P_{t,t}(s, k) \geqslant \frac{2 \cdot k \cdot s}{s+t}$, [5, Th. 3].

b) For any integer $t \geqslant 2$ and any integer $s > t$, $P_{t,t}(s, k) \geqslant \frac{k \cdot s \cdot (2s-2t+1)}{(2s-2t+1)t+(s-t)^2}$, [5, Th. 4].

c) For any integer $t \geqslant 2$ and any integer $s > 2t$, $P_{t,t}(s, k) \geqslant \frac{2k \cdot s \cdot (s+1)}{(s-t)^2+3st-t^2+2t}$, [50, Th. 16].

d) For any integer $t \geqslant 2$ and any integer $t < s \leqslant 2t$, $P_{t,t}(s, k) \leqslant \frac{k \cdot s \cdot (2s-2t+1)}{(2s-2t+1)t+(s-t)^2}$, [5, Th. 6].

e) For any integers p, t with $p \leqslant t + 1$, $P_{t,t}(pt, k) \leqslant m$, where $k = \binom{t}{t-p+1}\binom{s}{t}$ and $m = \binom{t}{t-p+1}\binom{s}{t} + \binom{s-p}{t-p+1}\binom{s-1}{p-1}$, [9, Th. 10].

TABLE I
SUMMARY OF PREVIOUS RESULTS

Code	s	k	t	ℓ	Lower bound	Upper bound	Notes and References
P	s	k	t	t	$\frac{2\cdot k\cdot s}{s+t}$		[5, Th. 3]
P	s	k	t	t	$\frac{k\cdot s\cdot(2s-2t+1)}{(2s-2t+1)t+(s-t)^2}$		$t\geqslant 2$ and $s>t$ [5, Th. 4]
P	s	k	t	t	$\frac{2k\cdot s\cdot(s+1)}{(s-t)^2+3st-t^2+2t}$		$t\geqslant 2$ and $s>2t$ [50, Th. 16]
P	s	k	t	t		$\frac{k\cdot s\cdot(2s-2t+1)}{(2s-2t+1)t+(s-t)^2}$	$t\geqslant 2$ and $t<s\leqslant 2t$ [5, Th. 6]
P	$p\cdot t$	$\binom{t}{t-p+1}\binom{s}{t}$	t	t		$\binom{t}{t-p+1}\binom{s}{t}+\binom{s-p}{t-p+1}\binom{s-1}{p-1}$	$p\leqslant t+1$ [9, Th. 10]
FP	$2t$	3	1	1	$3t+2$	$3t+2$	$t\geqslant 2$ [51]
FP	$2t$	4	1	1	$3t+3$	$3t+3$	$t\geqslant 2$ [51]
FP	$2t+1$	3	1	1	$3t+3$	$3t+4$	$t\geqslant 2$ [51]
FP	$2t$	4	1	1	$3t+4$	$3t+5$	$t\geqslant 2$ [51]

Note that for any two integers $t\geqslant 2$ and $s>t$, the bound in Theorem 2(b) improves upon the bound in Theorem 2(a). This is verified by showing that $\frac{k\cdot s\cdot(2s-2t+1)}{(2s-2t+1)t+(s-t)^2}-\frac{2\cdot k\cdot s}{s+t}\geqslant 0$ by basic algebraic manipulations. However the lower bound in Theorem 2(a) holds for all values of s, while the one in Theorem 2(b) only for $s>t$. Also, in [50] it was shown that for any two integers $t\geqslant 2$ and $s>2t$, the bound in Theorem 2(c) is stronger than the bound in Theorem 2(b).

The result in Theorem 2(d) is achieved by Construction 1 in [5]. The authors of [5] presented another construction which is not reported here due to its length. For the exact details please refer to [5, Construction 4 and Th.8]. This construction was then improved in [50] and in [9]. Several more constructions of PIR array codes have also been presented in [9], [50]. For the convenience of the reader we provide a summary of previous results in Table I.

The following theorem summarizes some of the known basic previous results, as well as several new ones. The proofs are rather simple and are thus omitted.

Theorem 3: For every s,k,t,ℓ,a positive integers:

a) $P_{t,\ell}(s,1)=B_{t,\ell}(s,1)=\lceil s/t\rceil$.
b) $FP_{t,\ell}(s,k_1+k_2)\leqslant FP_{t,\ell}(s,k_1)+FP_{t,\ell}(s,k_2)$ (also for P, B, and FB).
c) $FP_{t,\ell}(s,a\cdot k)\leqslant a\cdot FP_{t,\ell}(s,k)$ (also for P, B, and FB).
d) $FP_{t,\ell}(s_1+s_2,k)\leqslant FP_{t,\ell}(s_1,k)+FP_{t,\ell}(s_2,k)$ (also for P, B, and FB).
e) $FP_{t,\ell}(a\cdot s,k)\leqslant a\cdot FP_{t,\ell}(s,k)$ (also for P, B, and FB).
f) $FP_{t,\ell}(s,k)\leqslant a\cdot FP_{a\cdot t,\ell}(s,k)$ (also for P, B, and FB).

One of the simplest ways to construct array PIR and batch codes uses the Gadget Lemma, which was first proved in [22].

Lemma 4 (The Gadget Lemma): Let $\mathcal{C}$ be an $(s,k,m,1,1)$ batch code, then for any positive integer t there exists an $(ts,k,m,t,1)$ batch array code $\mathcal{C}'$ (denoted also by $t\cdot\mathcal{C}$).

It is easily verified that the Gadget Lemma holds also for PIR codes and therefore $P_{t,\ell}(s,k)\leqslant P_{t,1}(s,k)\leqslant P(\lceil s/t\rceil,k)$ and $B_{t,\ell}(s,k)\leqslant B_{t,1}(s,k)\leqslant B(\lceil s/t\rceil,k)$. However, unfortunately, the Gadget Lemma does not hold in general for functional PIR and batch codes. Even a weaker variation of the Gadget Lemma, where $\ell=t$, does not hold in general for functional PIR and batch codes either. Assume by contradiction that if there is an $(s,k,m,1,1)$ functional PIR code $\mathcal{C}$, then for any positive integer t there exists a (ts,k,m,t,t) functional PIR array code. Then, this will imply that $FP_{t,t}(ts,k)\leqslant FP(s,k)$. However, it is known that $FP(2,2)=3$ by the simple parity code. Thus, under this assumption it would hold that $FP_{2,2}(4,2)\leqslant FP(2,2)=3$. But, according to a lower bound on functional PIR array codes, which will be shown in Theorem 9, it holds that $FP_{2,2}(4,2)\geqslant\frac{2\cdot 2\cdot 15}{15+3}>3$, which is a contradiction.

IV. LOWER BOUNDS ON ARRAY CODES

In this section we present several lower bounds on functional PIR and batch array codes. Let $\left\{{a\atop b}\right\}$ be the Stirling number of the second kind, which calculates the number of partitions of a set of a elements into b nonempty subsets. It is well known that $\left\{{a\atop b}\right\}=\frac{1}{b!}\sum_{i=0}^{b}(-1)^{b-i}\binom{b}{i}i^a$.

Theorem 5: For all s,k,t and ℓ positive integers $FB_{t,\ell}(s,k)\geqslant m^*$, where m^* is the smallest positive integer such that

$$\sum_{i=k}^{m^*}\binom{m^*}{i}\cdot\left\{{i\atop k}\right\}\cdot\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^i\geqslant\binom{2^s+k-2}{k}.$$

Proof: Let $\mathcal{C}$ be an optimal (s,k,m^*,t,ℓ) functional batch array code. Since there are s information bits, there are (2^s-1) possible linear combination requests. Recall that by the definition of functional batch array codes, each multiset request has k mutually disjoint recovery sets of linear combinations. The number of possibilities to choose k elements from a set of 2^s-1 elements if repetitions are allowed is $\binom{2^s+k-2}{k}$. Thus,

there are $\binom{2^s+k-2}{k}$ possible multiset requests of length k. For each multiset request of k linear combinations $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ of the information bits, there is a partition of the buckets of the code $\mathcal{C}$ into k recovering sets $S_1, \ldots, S_k \subseteq [m^*]$ such that $\boldsymbol{v}_j, j \in [k]$ can be recovered by reading at most ℓ bits from each column in S_j.

In each bucket there are t cells where at most ℓ cells from them can be read. Thus, there are $\sum_{j=1}^{\ell}\binom{t}{j}$ nonzero linear combinations that can be obtained from one bucket. For any positive integer n, there are $(\sum_{j=1}^{\ell}\binom{t}{j})^n$ nonzero linear combinations that can be obtained from n buckets while using all the n buckets.

In order to satisfy a multiset request, the buckets must be divided into k disjoint recovering sets such that each set can satisfy one requested linear combination. There are

$$\sum_{i=k}^{m^*}\binom{m^*}{i}\cdot\left\{{i \atop k}\right\}$$

possibilities to divide at most m^* buckets into k nonempty disjoint sets. Each subset of the buckets of size at least k can be divided into k nonempty sets. Thus, we take the sum over all the subsets of the buckets of size at least k, where for each such subset we count the number of possibilities to divide it into k nonempty subsets using Stirling number of the second kind. From each subset of size p where $k \leqslant p \leqslant m^*$, there exist $(\sum_{j=1}^{\ell}\binom{t}{j})^p$ linear combinations. Therefore, for a given partition of $i, k \leqslant i \leqslant m^*$ buckets into k subsets such that the sizes of the subsets are $p_1, p_2, \ldots, p_k$ where $\sum_{j=1}^{k} p_j = i$, the number of different k-sets of linear combinations such that each linear combination taken from one subset is

$$\prod_{p\in\{p_1,p_2,\cdots,p_k\}}\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^p=\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^i.$$

In order to satisfy each multiset request by a set of k linear combinations such that each linear combination satisfies one requested linear combination. It must hold that the number of different k-sets of linear combinations such that each linear combination taken from one subset of the buckets, for all partitions of the m^* buckets into k nonempty disjoint subsets, is larger than the number of multiset requests. Thus,

$$\sum_{i=k}^{m^*}\binom{m^*}{i}\cdot\left\{{i \atop k}\right\}\cdot\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^i\geqslant\binom{2^s+k-2}{k}. \quad (1)$$

■

A similar lower bound can be obtained for functional PIR array codes. While in functional batch array codes there exist $\binom{2^s+k-2}{k}$ possible multiset requests, in functional PIR array codes there exist 2^s-1 possible requests.

Corollary 6: For all s, k, t and ℓ positive integers $FP_{t,\ell}(s,k) \geqslant m^*$, where m^* is the smallest positive integer such that

$$\sum_{i=k}^{m^*}\binom{m^*}{i}\cdot\left\{{i \atop k}\right\}\cdot\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^i\geqslant 2^s-1. \quad (2)$$

Another combinatorial bound for functional PIR array codes is shown in the following theorem.

Theorem 7: For all s, k, t and ℓ positive integers $FP_{t,\ell}(s,k) \geqslant m^*$, where m^* is the smallest positive integer such that

$$\sum_{i=1}^{m^*-k+1}\binom{m^*}{i}\cdot\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^i\geqslant k\cdot(2^s-1).$$

Proof: Let $\mathcal{C}$ be an optimal (s, k, m^*, t, ℓ) functional PIR array code. Since there are s information bits, there are (2^s-1) possible requests. The code $\mathcal{C}$ must satisfy each request k times by k linear combinations from k disjoint recovering sets. In other words, for each request there are k nonempty disjoint recovering sets, such that each set has a linear combination equal to the request. Each recovering set must be of size at most m^*-k+1, in order to have other $k-1$ nonempty recovering sets.

In each bucket there are t cells where at most ℓ cells from them can be read. Thus, there are $\sum_{i=1}^{\ell}\binom{t}{i}$ nonzero linear combinations that can be obtained from one bucket and $(\sum_{j=1}^{\ell}\binom{t}{j})^n$ from n buckets, for any positive integer n, while using all the n buckets. We are interested in counting the different linear combinations that can be obtained from at most m^*-k+1 buckets. Thus, there are

$$\sum_{i=1}^{m^*-k+1}\binom{m^*}{i}\cdot\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^i$$

such linear combinations. It must hold that the number of different linear combinations that can be got from at most m^*-k+1 buckets is larger than k times the number of the possible requests. Thus,

$$\sum_{i=1}^{m^*-k+1}\binom{m^*}{i}\cdot\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^i\geqslant k\cdot(2^s-1). \quad (3)$$

■

The following corollary is derived from Theorem 7.

Corollary 8: $FP_{t,\ell}(s,k)\geqslant\left\lceil\frac{\log_2(k(2^s-1)+1)}{\log_2(\sum_{i=0}^{\ell}\binom{t}{i})}\right\rceil$, for all s, k, t and ℓ positive integers.

Proof: The proof of Theorem 7 can be modified by using a weaker constraint, that the size of each subset is at most m. Thus, it must hold that $\sum_{i=1}^{m}\binom{m}{i}\cdot\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^i\geqslant k\cdot(2^s-1)$. From the equality $\sum_{i=0}^{m}\binom{m}{i}\cdot x^i=(x+1)^m$, we get that,

$$\begin{aligned}\sum_{i=1}^{m}\binom{m}{i}\cdot\left(\sum_{j=1}^{\ell}\binom{t}{j}\right)^i&=\left(1+\sum_{j=1}^{\ell}\binom{t}{j}\right)^m-1\\&=\left(\sum_{j=0}^{\ell}\binom{t}{j}\right)^m-1\geqslant k\cdot(2^s-1).\end{aligned}$$

Therefore, a lower bound over the minimal number of buckets, is $FP_{t,\ell}(s,k)\geqslant\left\lceil\frac{\log_2(k(2^s-1)+1)}{\log_2(\sum_{j=0}^{\ell}\binom{t}{j})}\right\rceil$. ■

Lastly in this section we show a different lower bound for functional PIR array codes, which is motivated

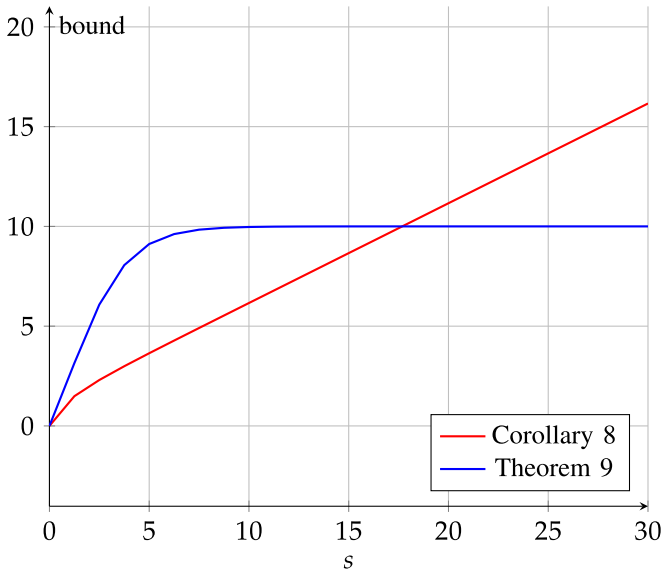

Fig. 1. Lower Bounds of Corollary 8 and Theorem 9 for $k = 5$ and $t = 2$.

by the corresponding lower bound for PIR array codes from [5, Th. 3].

Theorem 9: For any s, k, t and ℓ positive integers, $FP_{t,\ell}(s,k) \geqslant \frac{2 \cdot k \cdot (2^s-1)}{(2^s-1)+\sum_{i=1}^{\ell}\binom{t}{i}}$.

Proof: Suppose there exists an (s,k,m,t,ℓ) functional PIR array code. There are 2^s-1 possible linear combination requests which are denoted by $\boldsymbol{u}_i$ for $1 \leqslant i \leqslant 2^s-1$. For $i \in [2^s-1]$, we define by α_i to be the number of recovering sets of size 1 of the i-th linear combination request $\boldsymbol{u}_i$.

Since it is possible to read at most ℓ bits from each bucket, every bucket can satisfy at most $\sum_{i=1}^{\ell}\binom{t}{i}$ linear combinations. Thus, the number of recovering sets of size 1 is $m \cdot \sum_{i=1}^{\ell}\binom{t}{i}$, and $\sum_{j=1}^{2^s-1}\alpha_j \leqslant m \cdot \sum_{i=1}^{\ell}\binom{t}{i}$. Hence, there exists $q \in [2^s-1]$ such that $\alpha_q \leqslant \frac{m \cdot \sum_{i=1}^{\ell}\binom{t}{i}}{2^s-1}$, so out of its k disjoint recovering sets of $\boldsymbol{u}_q$, at most α_q of them are of size 1, and the size of each of the remaining $k-\alpha_q$ subsets is at least 2. Hence,

$$m \geqslant \alpha_q + 2(k-\alpha_q) = 2k - \alpha_q \geqslant 2k - \frac{m \cdot \sum_{i=1}^{\ell}\binom{t}{i}}{2^s-1},$$

and therefore $m(1+\frac{\sum_{i=1}^{\ell}\binom{t}{i}}{(2^s-1)}) \geqslant 2k$, which implies that $FP_{t,\ell}(s,k) \geqslant \frac{2k(2^s-1)}{(2^s-1)+\sum_{i=1}^{\ell}\binom{t}{i}}$. ∎

Figure 1 compares between the two lower bounds of Corollary 8 and Theorem 9. In this comparison we specify the parameters $k = 5$ and $t = 2$. We can see that for $0 \leqslant s < 18$ the lower bound of Theorem 9 is stronger, but for $s \geqslant 18$ the lower bound of Corollary 8 is stronger.

V. General Constructions of Array Codes

In this section we present several constructions of array codes for functional PIR and batch codes.

A. Basic Constructions

Even though the Gadget Lemma cannot be extended in general for functional PIR and batch codes, here we show a variation of it that will hold. For any positive integer i, $\mathbf{0}^i$ denotes the zero vector of length i, and for any two vectors $\boldsymbol{v}$ and $\boldsymbol{u}$, the vector $\boldsymbol{vu}$ is defined to be the concatenation of $\boldsymbol{u}$ after $\boldsymbol{v}$.

Lemma 10: For any positive integer p, if there exists an $(s, p \cdot k, m, t, \ell)$ functional batch array code, then there exists an $(p \cdot s, k, m, p \cdot t, \ell)$ functional batch array code. Therefore,

$$FP_{p\cdot t,\ell}(p \cdot s, k) \leqslant FB_{p\cdot t,\ell}(p \cdot s, k) \leqslant FB_{t,\ell}(s, p \cdot k),$$

and in particular, $FP_{t,1}(s,k) \leqslant FB_{t,1}(s,k) \leqslant FB(\lceil \frac{s}{t} \rceil, t \cdot k)$.

Proof: Let $\mathcal{C}$ be an $(s, p \cdot k, m, t, \ell)$ functional batch array code with encoding function $\mathcal{E}$ and decoding function $\mathcal{D}$. We construct an $(p \cdot s, k, m, p \cdot t, \ell)$ functional batch array code $\mathcal{C}'$ by using the code $\mathcal{C}$. Let $\mathcal{S} = \{x_{i,j} : 1 \leqslant i \leqslant p, 1 \leqslant j \leqslant s\}$ be the set of $p \cdot s$ information bits. The $p \cdot s$ information bits can be partitioned into p parts, each of size s, such that part $i, i \in [p]$ is $\mathcal{S}_i = \{x_{i,j} : 1 \leqslant j \leqslant s\}$. The code $\mathcal{C}'$ will be represented by a $pt \times m$ array A, that contains p subarrays $A_1, A_2, \ldots, A_p$ each of dimension $t \times m$. In the encoding function of the code $\mathcal{C}'$, the i-th subarray A_i stores the encoded bits of the set $\mathcal{S}_i$ by applying the encoding function $\mathcal{E}$ of the code $\mathcal{C}$ over the information bits in the set $\mathcal{S}_i$.

Let $R = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k\}$ be a multiset request of size k of the $p \cdot s$ information bits, where $\boldsymbol{v}_i, i \in [k]$ is a binary vector of length $p \cdot s$ that represents the i-th request, which is a linear combination of the $p \cdot s$ information bits. For each $i \in [k]$, denote $\boldsymbol{v}_i = (\boldsymbol{v}_i^1, \boldsymbol{v}_i^2, \ldots, \boldsymbol{v}_i^p)$ where $\boldsymbol{v}_i^j, j \in [p]$ is a vector of length s that represents the linear combination of the bits in $\mathcal{S}_j$. Let $R^* = \{\boldsymbol{v}_i^j : 1 \leqslant i \leqslant k, 1 \leqslant j \leqslant p\}$ be a multiset request of size pk, that has pk vectors of length s each. By using the decoding function $\mathcal{D}$ of the code $\mathcal{C}$ with the request R^* we get pk recovering sets. For each $i \in [k]$ and $j \in [p]$, let $B_i^j = \{(h_{i,1}, \boldsymbol{u}_{i,1}), (h_{i,2}, \boldsymbol{u}_{i,2}), \ldots, (h_{i,a_i}, \boldsymbol{u}_{i,a_i})\}$ be a recovering set for $\boldsymbol{v}_i^j$ of size a_i, where for each $g \in [a_i]$, $(h_{i,g}, \boldsymbol{u}_{i,g})$ is a pair of a bucket $h_{i,g}$ with a vector $\boldsymbol{u}_{i,g}$ of length t that indicates the cells which are read from the bucket $h_{i,g}$. For each B_i^j and $f \in [p]$, let $B_{i,f}^j = \{(h_{i,1}, \mathbf{0}^{t(f-1)}\boldsymbol{u}_{i,1}\mathbf{0}^{t(p-f)}), \ldots, (h_{i,a_i}, \mathbf{0}^{t(f-1)}\boldsymbol{u}_{i,a_i}\mathbf{0}^{t(p-f)})\}$ be a recovering set for $\boldsymbol{v}_i^j$, that reads the cells of subarray A_f. For each $i \in [k]$, to satisfy the request $\boldsymbol{v}_i$, the union $\cup_{f=1}^{p} B_{i,f}^f$ is taken, since for each $f \in [p]$ the subset $B_{i,f}^f$ can satisfy the request $\boldsymbol{v}_i^f$.

For each $f_1, f_2 \in [p]$, $i_1, i_2 \in [k]$ and $j_1, j_2 \in [p]$, $B_{i_1,f_1}^{j_1}$ and $B_{i_2,f_2}^{j_2}$ have disjoint subsets of buckets if $i_1 \neq i_2$ or $j_1 \neq j_2$, because $B_{i_1}^{j_1}$ and $B_{i_2}^{j_2}$ have disjoint subsets of buckets if $i_1 \neq i_2$ or $j_1 \neq j_2$. Thus, for any $i \neq j \in [k]$, $\cup_{f=1}^{p} B_{i,f}^f$ and $\cup_{f=1}^{p} B_{j,f}^f$ have disjoint subsets of buckets.

It remains to show that we read at most ℓ cells from each bucket. Recall that for any $\boldsymbol{v}_i, i \in [k]$ the union $\cup_{f=1}^{p} B_{i,f}^f$ satisfies the request $\boldsymbol{v}_i$. Thus, if the recovering set B_{i,f_1}^j was used then $f_1 = j$. Therefore, the recovering set B_{i,f_2}^j for each $f_2 \neq f_1$ was not used. From the previous paragraph we showed that $B_{i,f_1}^{j_1}$ and $B_{i,f_2}^{j_2}$ have disjoint subsets of buckets if $j_1 \neq j_2$. Thus, the recovering sets that were used to satisfy $\boldsymbol{v}_i$ have disjoint subsets of buckets. Thus, each bucket can appear in at most one of these recovering sets, and it is known that each one of these subsets uses at most ℓ cells from each bucket from the properties of the code $\mathcal{C}$.

The last claim in the lemma holds by setting $p = t$ and $t = 1$. ■

Another general construction is stated in the next theorem.

Theorem 11: For any positive integers, s, k, t, t_0, and ℓ, $FB_{t,\ell}(s,k) \leqslant m + m_0$, where $m = FB_{t+t_0,\ell}(s,k)$ and $m_0 = FB_{t,\ell}(m \cdot t_0, k)$.

Proof: Let $\mathcal{C}_1, \mathcal{C}_2$ be an $(s, k, m, t + t_0, \ell), (m \cdot t_0, k, m_0, t, \ell)$ functional batch array code, respectively. We construct an $(s, k, m+m_0, t, \ell)$ functional batch array code $\mathcal{C}$ by using the codes $\mathcal{C}_1, \mathcal{C}_2$. First, the s information bits are encoded using the encoder function of the code $\mathcal{C}_1$ to get a $(t + t_0) \times m$ array A. Then, the $t_0 \cdot m$ bits in the last t_0 rows of A are encoded into a $t \times m_0$ array B using the encoder function of the code $\mathcal{C}_2$. The code $\mathcal{C}$ will be represented by a $t \times (m + m_0)$ array, where the first m buckets (columns) will be the first t rows of the array A and the last m_0 buckets will be the array B.

Let $R = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\}$ be a multiset request of size k, where $\boldsymbol{v}_i, i \in [k]$ is a binary vector of length s that represents the i-th request. Denote by $\{E_1, \ldots, E_k\}$ the k recovering sets that are obtained by using the decoding function of the code $\mathcal{C}_1$ with the request R. For each $i \in [k]$, assume that $|E_i| = p_i$ and denote $E_i = \{(h_{i,1}, \boldsymbol{u}_{i,1}), \ldots, (h_{i,p_i}, \boldsymbol{u}_{i,p_i})\}$ where for each $j \in [p_i]$, $(h_{i,j}, \boldsymbol{u}_{i,j})$ is a pair of a bucket $h_{i,j}$ with a vector $\boldsymbol{u}_{i,j}$ of length $t + t_0$ that indicates the cells which are read from the bucket $h_{i,j}$. For each $i \in [k]$ and $j \in [p_i]$, let $\boldsymbol{u}'_{i,j}$ be the vector with the last t_0 entries of $\boldsymbol{u}_{i,j}$ and let $R'_{i,j}$ be the sum of the bits in the cells that indicated by $\boldsymbol{u}'_{i,j}$.

Let $R' = \{\sum_{j=1}^{p_1} R'_{1,j}, \ldots, \sum_{j=1}^{p_k} R'_{k,j}\}$ be a multiset request of size k. Denote by $\{F_1, \ldots, F_k\}$ the k recovering sets that are obtained by using the decoding function of the code $\mathcal{C}_2$ with the multiset request R'. To satisfy $\boldsymbol{v}_i$, the code $\mathcal{C}$ can use the recovering set $F_i \cup E'_i$, where $E'_i = \{(h_{i,1}, \boldsymbol{u}''_{i,1}), \ldots, (h_{i,k}, \boldsymbol{u}''_{i,k})\}$ where for each $j \in [k]$, $\boldsymbol{u}''_{i,j}$ is the vector with the first t entries of $\boldsymbol{u}_{i,j}$.

It remains to show that at most ℓ cells are read from each bucket. Each $\boldsymbol{v}_i, i \in [k]$ has a recovering set $F_i \cup E'_i$, where the recovering set F_i of $\mathcal{C}_2$ uses at most ℓ cells from each bucket from the property of the code $\mathcal{C}_2$. Also, the recovering set E_i of $\mathcal{C}_1$ uses at most ℓ cells from each bucket from the property of the code $\mathcal{C}_1$. Thus, E'_i also uses at most ℓ cells. ■

Note that a similar statement can hold for functional PIR array code, where for any positive integers s, k, t, t_0, and ℓ, $FP_{t,\ell}(s,k) \leqslant m + m_0$, where $m = FP_{t+t_0,\ell}(s,k)$ and $m_0 = FB_{t,\ell}(m \cdot t_0, k)$.

Example 1: In this example we demonstrate the construction of a $(12, 1, 4, 4, 2)$ functional batch array code according to Theorem 11 by using a $(12, 1, 3, 5, 2)$ functional batch array code as in Table II and a $(3, 1, 1, 4, 2)$ functional batch array code as in Table III. The construction is given in Table IV. It can be verified that $FB_{4,2}(12, 1) = 4$. Note that in this example and in the rest of the paper the notation $x_{i_1} x_{i_2} \cdots x_{i_h}$ is a shorthand to the summation $x_{i_1} + x_{i_2} + \cdots + x_{i_h}$.

TABLE II
$(12, 1, 3, 5, 2)$ FUNCTIONAL BATCH ARRAY CODE

1	2	3
x_1	x_5	x_9
x_2	x_6	x_{10}
x_3	x_7	x_{11}
x_4	x_8	x_{12}
$x_1x_2x_3x_4$	$x_5x_6x_7x_8$	$x_9x_{10}x_{11}x_{12}$

TABLE III
$(3, 1, 1, 4, 2)$ FUNCTIONAL BATCH ARRAY CODE

1
x_1
x_2
x_3
$x_1x_2x_3$

TABLE IV
$(12, 1, 4, 4, 2)$ FUNCTIONAL PIR/BATCH ARRAY CODE

1	2	3	4
x_1	x_5	x_9	$x_1x_2x_3x_4$
x_2	x_6	x_{10}	$x_5x_6x_7x_8$
x_3	x_7	x_{11}	$x_9x_{10}x_{11}x_{12}$
x_4	x_8	x_{12}	$x_1x_2\cdots x_{12}$

B. Constructions Based Upon Covering Codes

In this section it is shown how covering codes are used to construct array codes. Denote by $d_H(\boldsymbol{x}, \boldsymbol{y})$ the Hamming distance between two vectors $\boldsymbol{x}, \boldsymbol{y}$, and denote by $w_H(\boldsymbol{x})$ the Hamming weight of $\boldsymbol{x}$. Also define $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ as the inner product of the two vectors $\boldsymbol{x}, \boldsymbol{y}$. Next we remind the definition of covering codes [12].

Definition 12: Let $n \geqslant 1$, $R \geqslant 0$ be integers. A code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is called an **R-covering code** if for every word $\boldsymbol{y} \in \mathbb{F}_q^n$ there is a codeword $\boldsymbol{x} \in \mathcal{C}$ such that $d_H(\boldsymbol{x}, \boldsymbol{y}) \leqslant R$. The notation $[n, k, R]_q$ denotes a linear code over $\mathbb{F}_q$ of length n, dimension k, and covering radius R. The value $g[n, R]_q$ denotes the smallest dimension of a linear code over $\mathbb{F}_q$ with length n and covering radius R. The value $h[s, R]_q$ is the smallest length of a linear code over $\mathbb{F}_q$ with covering radius R and redundancy s. In case $q = 2$ we will remove it from these notations.

The following property is well known for linear covering codes; see e.g. [12, Th. 2.1.9].

Property 13: For an $[n, k, R]$ linear covering code with some parity check matrix H, every syndrome vector $s \in \Sigma^{n-k}$ can be represented as the sum of at most R columns of H.

The connection between linear codes and functional batch array codes is established in the next theorem.

Theorem 14: Let $\mathcal{C}$ be a $[t, t - s, \ell]$ linear covering code. Then, there exists an $(s, 1, 1, t, \ell)$ functional batch array code. In particular, $FB_{t,\ell}(t - g[t, \ell], 1) = 1$.

Proof: Let $\boldsymbol{x} = (x_1, \ldots, x_s)$ the vector of dimension $1 \times s$ with the s information bits, and let H be a parity check matrix of the code $\mathcal{C}$, with dimension $s \times t$. We construct an $(s, 1, 1, t, \ell)$ functional batch array code $\mathcal{C}'$ by taking each entry of the vector $\boldsymbol{c} = (\boldsymbol{x}H)^\intercal$ as a cell in the code. The dimension of $\boldsymbol{c}$ is $t \times 1$, and thus, we get one bucket with t cells where each cell has a linear combination of the s information bits.

Let $\boldsymbol{u} \in \Sigma^s$ be a request which represents the linear combination $\langle \boldsymbol{u}, \boldsymbol{x} \rangle$ of the s information bits. From Property 13, we know that there exists a vector $\boldsymbol{y} \in \Sigma^t$ such that $\boldsymbol{y} \cdot H^\intercal = \boldsymbol{u}$, where $w = w_H(\boldsymbol{y}) \leqslant \ell$. Let $\mathcal{A} = \{i : i \in [t], y_i = 1\}$,

where y_i is the entry number i of $\boldsymbol{y}$. Thus, $\langle \boldsymbol{u}, \boldsymbol{x} \rangle = \boldsymbol{u} \cdot \boldsymbol{x}^\intercal = \boldsymbol{y} \cdot H^\intercal \cdot \boldsymbol{x}^\intercal = \boldsymbol{y} \cdot \boldsymbol{c} = \sum_{i \in \mathcal{A}} c_i$, where c_i is the entry number i of $\boldsymbol{c}$. Therefore, to satisfy the request $\langle \boldsymbol{u}, \boldsymbol{x} \rangle$ we should read $|\mathcal{A}| = w \leqslant \ell$ cells from the code $\mathcal{C}'$.

Recall that $g[t, \ell]$ is the smallest dimension of a linear code with length t and covering radius ℓ. Thus, there exists a $[t, g[t, \ell], \ell]$ linear covering code. We get that there exists a $(t - g[t, \ell], 1, 1, t, \ell)$ functional batch array code, which implies that $FB_{t,\ell}(t - g[t, \ell], 1) = 1$. ∎

Theorem 14 holds also for functional PIR array code and thus the following results are derived.

Corollary 15: Let s, k, t and ℓ be positive integers. Then,

a) $FP_{t,\ell}(s, k) \leqslant FB_{t,\ell}(s, k) \leqslant k \cdot \left\lceil \frac{s}{t - g[t,\ell]} \right\rceil$.
b) $FP_{t+t_0,\ell}(s, k) \leqslant FP_{t,t}(s, k)$, where $t_0 = g[t + t_0, \ell]$. Also works for FB.
c) $FP_{t,\ell}(s, k) \leqslant FB_{t,\ell}(s, k) \leqslant k \cdot \left(\left\lceil \frac{s}{\alpha} \right\rceil + 1\right)$, where $\left\lceil \frac{s}{\alpha} \right\rceil \leqslant t - g[t, \ell]$, and $\alpha = (t + 1) - g[(t + 1), \ell]$.

Proof:

a) From Theorem 3(c) we get that $FB_{t,\ell}(s, k) \leqslant k \cdot FB_{t,\ell}(s, 1) \leqslant k \cdot FB_{t,\ell}(\frac{s}{t - g[t,\ell]} \cdot (t - g[t, \ell]), 1)$. In addition, from Theorem 3(e) we get that $k \cdot FB_{t,\ell}(\frac{s}{t - g[t,\ell]} \cdot (t - g[t, \ell]), 1) \leqslant k \cdot \frac{s}{t - g[t,\ell]} \cdot FB_{t,\ell}(t - g[t, \ell], 1)$ and from Theorem 14 it holds that $FB_{t,\ell}(t - g[t, \ell], 1) \leqslant 1$. Thus, $FB_{t,\ell}(s, k) \leqslant k \cdot \left\lceil \frac{s}{t - g[t,\ell]} \right\rceil$.
b) Let the code $\mathcal{C}$ be an (s, k, m, t, t) functional PIR array code. From each bucket, which has at most t cells, we construct a new bucket with at most $t + t_0$ cells to get any possible linear combination of the t cells in the appropriate bucket. This is possible by taking the linear combinations in the cells as the information bits and using Theorem 14 where it holds that $FP_{t+t_0}(t, 1) = FP_{t+t_0}(t + t_0 - g[t + t_0], 1) = 1$. Given a request R that the code must satisfy with k disjoint recovering sets by reading at most ℓ cells from each bucket. The same recovering sets for R as in the code $\mathcal{C}$ can be taken while reading at most ℓ cells from each bucket in the new code in order to get the needed linear combination. Thus, we get an $(s, k, m, t + t_0, \ell)$ functional PIR array code, and hence, $FP_{t+t_0,\ell}(s, k) \leqslant FP_{t,t}(s, k)$.
c) From Theorem 11 we can get that $FB_{t,\ell}(s, 1) \leqslant m + FB_{t,\ell}(m, 1)$, where $m = FB_{t+1,\ell}(s, k)$. From the first claim in this corollary we know that $m = FB_{t+1,\ell}(s, k) \leqslant \left\lceil \frac{s}{(t+1) - g[(t+1),\ell]} \right\rceil \leqslant t - g[t, \ell]$. Thus, from Theorem 14 we get that $FB_{t,\ell}(m, 1) = 1$. Therefore, $FB_{t,\ell}(s, 1) \leqslant m + FB_{t,\ell}(m, 1) \leqslant \left\lceil \frac{s}{(t+1) - g[(t+1),\ell]} \right\rceil + 1$. According to Theorem 3(e) we get that $FB_{t,\ell}(s, k) \leqslant k \cdot FB_{t,\ell}(s, 1) \leqslant k \cdot (\lceil \frac{s}{\alpha} \rceil + 1)$ where $\alpha = (t + 1) - g[(t + 1), \ell]$.

∎

C. The Cases of $k = 1, 2$

Even though the cases of $k = 1, 2$ are the most trivial ones when the codewords are vectors, they are apparently not easily solved for array codes. In this section we summarize some of our findings on these important and interesting cases.

Theorem 16: For each s, t, ℓ positive integers:

a) $FP_{t,\ell}(s, 1) \geqslant \left\lceil \frac{s}{\log_2\left(\sum_{i=0}^{\ell} \binom{t}{i}\right)} \right\rceil$.
b) $FP_{t,t}(s, 1) = \lceil \frac{s}{t} \rceil$.
c) $FP_{t,1}(\lfloor \log_2(t + 1) \rfloor, 1) = 1$ and $\left\lceil \frac{s}{\log_2(t+1)} \right\rceil \leqslant FP_{t,1}(s, 1) \leqslant \left\lceil \frac{s}{\lfloor \log_2(t+1) \rfloor} \right\rceil$.
d) $FP_{t,\alpha \cdot t}(s, 1) \leqslant \left\lceil \frac{s}{t - g[t, \alpha \cdot t]} \right\rceil$, where $0 < \alpha < 1$.
e) $FP_{t,t/2}(s, 1) = \frac{s}{t} + 1$, where t is even, $\frac{s}{t}$ is integer, and $\frac{s}{t} \leqslant t - 1$.

Proof:

a) From corollary 8.
b) The lower bound over $FP_{t,t}(s, 1)$ is obtained by using the lower bound from the first claim of this theorem, $FP_{t,t}(s, 1) \geqslant \left\lceil \frac{s}{\log_2\left(\sum_{i=0}^{t} \binom{t}{i}\right)} \right\rceil = \lceil \frac{s}{t} \rceil$. The upper bound can be verified by showing that there exists an $(s, 1, \lceil \frac{s}{t} \rceil, t, t)$ functional PIR array code. There are t cells in each buckets. Then, in order to write all the s information bits there is a need to $\lceil \frac{s}{t} \rceil$ buckets. Each request is a linear combination of the s information bits. Thus, each request can be satisfied by reading the information bits which included in the request. It was shown that $FP_{t,t}(s, 1) \geqslant \lceil \frac{s}{t} \rceil$ and there exists an $(s, 1, m, t, t)$ functional PIR array code. Therefore, $FP_{t,t}(s, 1) = \lceil \frac{s}{t} \rceil$.
c) A $(\lfloor \log_2(t + 1) \rfloor, 1, 1, t, 1)$ functional PIR array code $\mathcal{C}$ can be obtained by writing all the $2^{\lfloor \log_2(t+1) \rfloor - 1} \leqslant t$ linear combinations of the information bits in at most t cells of one bucket. Each request is a linear combination of the information bits, and hence, for each request there exists a cell in the bucket that satisfies it. Thus, the appropriate cell can satisfy the request. The minimum number of buckets is 1. Thus, $FP_{t,1}(\lfloor \log_2(t+1) \rfloor, 1) = 1$. The lower bound over $FP_{t,1}(s, 1)$ is derived from the first claim of this theorem. Thus $FP_{t,1}(s, 1) \geqslant \left\lceil \frac{s}{\log_2(\sum_{i=0}^{1} \binom{t}{i})} \right\rceil = \left\lceil \frac{s}{\log_2(t+1)} \right\rceil$. The upper bound is shown by using Theorem 3(e),

$$\begin{aligned} FP_{t,1}(s, 1) &= FP_{t,1}\left(\frac{s}{\lfloor \log_2(t+1) \rfloor} \cdot \lfloor \log_2(t+1) \rfloor, 1\right) \\ &\leqslant FP_{t,1}\left(\left\lceil \frac{s}{\lfloor \log_2(t+1) \rfloor} \right\rceil \cdot \lfloor \log_2(t+1) \rfloor, 1\right) \\ &\leqslant \left\lceil \frac{s}{\lfloor \log_2(t+1) \rfloor} \right\rceil \cdot FP_{t,1}\left(\lfloor \log_2(t+1) \rfloor, 1\right) \\ &\leqslant \left\lceil \frac{s}{\lfloor \log_2(t+1) \rfloor} \right\rceil. \end{aligned}$$

d) From Corollary 15(a).
e) The lower bound over $FP_{t,t/2}(s, 1)$ can be found using the lower bound from the first claim of this theorem,

$$\begin{aligned} FP_{t,t/2}(s, 1) &\geqslant \left\lceil \frac{s}{\log_2(\sum_{i=0}^{t/2} \binom{t}{i})} \right\rceil \\ &\geqslant \left\lceil \frac{s}{\log_2(\sum_{i=0}^{t} \binom{t}{i})} \right\rceil + 1 \\ &= \left\lceil \frac{s}{t} \right\rceil + 1. \end{aligned}$$

For the upper bound, from Corollary 15(c) we get that $FP_{t,t/2}(s,1) \leqslant \left\lceil \frac{s}{(t+1)-g[(t+1),t/2]} \right\rceil + 1$. Since $g[t+1,t/2]=1$, then $FP_{t,t/2}(s,1) \leqslant s/t+1$. Lastly we need to show that $\left\lceil \frac{s}{(t+1)-g[(t+1),t/2]} \right\rceil \leqslant t - g[t,\ell]$ in order to use Corollary 15(c). Since $s/t \leqslant t-1$, it is derived that $\left\lceil \frac{s}{(t+1)-g[(t+1),t/2]} \right\rceil = \frac{s}{t} \leqslant t-1 = t - g[t+1,t/2] = g[t,t/2]$. Thus, $FP_{t,t/2} = \frac{s}{t}+1$. ∎

Example 2: In this example we demonstrate the construction of a $(12,1,4,4,2)$ functional PIR array code according to Theorem 16(e). The construction is given in Table IV. It can be verified that $FP_{4,2}(12,1)=4$.

An improvement for the case of $\ell = 1$ is proved in the following theorem.

Theorem 17: For any positive integers s_1, s_2, and t,

$$FP_{t,1}(s_1+s_2,1) \leqslant \left\lceil \frac{s_1}{\lfloor \log_2(t+1) \rfloor} \right\rceil + 1,$$

where $2^{s_2} - 1 \leqslant \left(\left\lceil \frac{s_1}{\lfloor \log_2(t+1) \rfloor} \right\rceil + 1 \right) (t - (2^{\lfloor \log_2(t+1) \rfloor} - 1))$.

Proof: A construction of an $(s_1+s_2,1,m,t,1)$ functional PIR array code for $m = \left\lceil \frac{s_1}{\lfloor \log_2(t+1) \rfloor} \right\rceil + 1$ is presented. The first s_1 information bits are divided into $m-1$ parts, where $h_i, i \in [m-1]$ is the size of part i, and $h_i \leqslant \lfloor \log_2(t+1) \rfloor$. Then, all the linear combinations of part $i \in [m-1]$ are written in the i-th bucket, so in each of the first $m-1$ buckets there are at least $t - (2^{\lfloor \log_2(t+1) \rfloor} - 1)$ empty cells. In the last bucket, the parity of each of the first $2^{\lfloor \log_2(t+1) \rfloor} - 1$ rows is stored. Since $2^{s_2} - 1 \leqslant m \cdot (t - (2^{\lfloor \log_2(t+1) \rfloor} - 1))$, each of the $2^{s_2}-1$ linear combinations of the s_2 bits can be written in the empty cells of the m buckets.

Let $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ be a request such that for any $i \in [m-1]$ the length of $\boldsymbol{v}_i$ is h_i, the length of $\boldsymbol{v}_m$ is s_2, and for simplicity assume that they are all nonzero. The linear combination $\boldsymbol{v}_m$ is satisfied by the cell where it is stored and assume it is in the j-th bucket, where $j < m$. Assume that the cell in the j-th bucket where the linear combination $\boldsymbol{v}_j$ is stored is in row r. We read from each bucket $b \in [m-1]$, where $b \neq j$ the cell with the linear combination represented by $\boldsymbol{v}_b + \boldsymbol{u}_b$, where $\boldsymbol{u}_b$ is the vector that represents the cell in bucket b in row r, but if $\boldsymbol{v}_b + \boldsymbol{u}_b = \boldsymbol{0}$ do not read from bucket b. Also, we read the cell in row r from the last bucket. Then, the obtained linear combination is the combination that is represented by $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1})$, because $\sum_{1 \leqslant b \leqslant m, b \neq j} \boldsymbol{u}_b = \boldsymbol{v}_j$ and for each $b \in [m-1]$ where $b \neq j$ we read the linear combination that is represented by $\boldsymbol{v}_b + \boldsymbol{u}_b$ from bucket b. ∎

For any t, s_1, s_2 where $s = s_1 + s_2$ and $s_2 \geqslant \lfloor \log_2(t+1) \rfloor$, the upper bound in Theorem 17 improves upon the one in Theorem 16(c) since $\left\lceil \frac{s}{\lfloor \log_2(t+1) \rfloor} \right\rceil \geqslant \left\lceil \frac{s_1}{\lfloor \log_2(t+1) \rfloor} \right\rceil + 1$.

Example 3: In this example the construction of a $(15,1,7,4,1)$ functional PIR array code is demonstrated based on Theorem 17. It can be verified that the parameters $t=4, s_1=12$ and $s_2=3$ satisfy the constraints of Theorem 17. The construction is given in Table V. The first $s_1 = 12$ information bits are partitioned into 6 parts, each part of size 2. All the nonzero linear combinations of part $i, i \in [6]$ are written in the i-th bucket with one cell remains empty. The sum of each of the first 3 rows is written. Now, there are still 7 empty cells, which are used to store all the nonzero linear combinations of the last $s_2 = 3$ bits in the empty cells. It can be concluded that $FP_{4,1}(15,1) \leqslant 7$, and from Theorem 16(c) we get that $FP_{4,1}(15,1) \geqslant 7$. Thus, $FP_{4,1}(15,1) = 7$.

TABLE V

$(15,1,7,4,1)$ FUNCTIONAL PIR ARRAY CODE

1	2	3	4	5	6	7
x_1	x_3	x_5	x_7	x_9	x_{11}	$x_1x_3x_5x_7x_9x_{11}$
x_2	x_4	x_6	x_8	x_{10}	x_{12}	$x_2x_4x_6x_8x_{10}x_{12}$
x_1x_2	x_3x_4	x_5x_6	x_7x_8	x_9x_{10}	$x_{11}x_{12}$	$x_1 \cdots x_{12}$
x_{13}	x_{14}	x_{15}	$x_{13}x_{14}$	$x_{13}x_{15}$	$x_{14}x_{15}$	$x_{13}x_{14}x_{15}$

TABLE VI

$(8,2,7,2,2)$ FUNCTIONAL PIR ARRAY CODE

1	2	3	4	5	6	7
x_1	x_2	x_1x_2	x_5	x_6	x_5x_6	$x_1x_2x_5x_6$
x_3	x_4	x_3x_4	x_7	x_8	x_7x_8	$x_3x_4x_7x_8$

Lastly, we report on several results for $k=2$.

Theorem 18: $6 \leqslant FB_{2,2}(8,2) \leqslant 7$.

Proof: The lower bound is obtained from Theorem 5. The upper bound is verified using the construction which appears in Table VI, i.e., the construction gives an $(8,2,7,2,2)$ functional batch array code. There are 8 information bits, 7 buckets, each one with 2 cells, and we show that this code can satisfy each multiset request of size 2. Let $\mathcal{S}_1 = \{x_1, x_2, x_3, x_4\}$ be a set of the first 4 information bits and $\mathcal{S}_2 = \{x_5, x_6, x_7, x_8\}$ be a set of the last 4 information bits. Let $R = \{\boldsymbol{v}_1, \boldsymbol{v}_2\}$ be a multiset request of size 2, where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are vectors of size 8. For each $i \in [2]$, $\boldsymbol{v}_i = (\boldsymbol{v}_i^1, \boldsymbol{v}_i^2)$ where $\boldsymbol{v}_i^j, j \in [2]$ is a vector of length 4 that represents a linear combination of the bits in $\mathcal{S}_j$. The possible linear combinations of $\mathcal{S}_1$ are divided into four different types in the following way.

a) The first type $\mathcal{T}_1$ includes the vectors that can be satisfied by using only one bucket from the buckets $1-3$.
b) The second type $\mathcal{T}_2$ includes any vector $\boldsymbol{u}$ that satisfies the following constraint. The vectors $\boldsymbol{u}+(1,1,0,0)$ and $\boldsymbol{u}+(0,0,1,1)$ can be satisfied by one bucket from buckets $1-3$. (The vector (1,1,0,0) represents the linear combination $x_1 + x_2$.)
c) The third type $\mathcal{T}_3$ includes any vector $\boldsymbol{u}$ that satisfies the following constraint. The vectors $\boldsymbol{u}+(1,1,1,1)$ and $\boldsymbol{u}+(1,1,0,0)$ can be satisfied by one bucket from the buckets $1-3$.
d) The fourth type $\mathcal{T}_4$ includes any vector $\boldsymbol{u}$ that satisfies the following constraint. The vectors $\boldsymbol{u}+(1,1,1,1)$ and $\boldsymbol{u}+(0,0,1,1)$ can be satisfied by one bucket from the buckets $1-3$.

These four types are disjoint and their union covers all the nonzero linear combinations of $\mathcal{S}_1$. From the symmetry of the first four information bits and the last four bits, the linear combinations of $\mathcal{S}_2$ are divided in the same way. It is possible to see that every two buckets from buckets $1-3$ can satisfy each possible linear combination of the first four bits. In the same way, every two buckets from buckets $4-6$ can satisfy each possible linear combination of the last four bits. Also,

the last bucket can satisfy each vector $(\boldsymbol{u},\boldsymbol{u})$, where $\boldsymbol{u} \in \{(1,1,0,0),(0,0,1,1),(1,1,1,1)\}$.

If one of the vectors $\{\boldsymbol{v}_1^1,\boldsymbol{v}_2^1\}$ is included in $\mathcal{T}_1$ (assume it is $\boldsymbol{v}_1^1$) and one of the vectors $\{\boldsymbol{v}_1^2,\boldsymbol{v}_2^2\}$ is included in $\mathcal{T}_1$ (assume it is $\boldsymbol{v}_1^2$), then these two vectors can be satisfied by one bucket from $1-3$ and one bucket from $4-6$. Then the remaining two buckets of $1-3$ can satisfy $\boldsymbol{v}_2^1$ and the remaining two buckets of $4-6$ can satisfy $\boldsymbol{v}_2^2$. Therefore, in this case the request R is satisfied by disjoint sets.

If there exist $2 \leqslant q_1, q_2 \leqslant 4$ where $\boldsymbol{v}_1^1 \in \mathcal{T}_{q_1}$ and $\boldsymbol{v}_1^2 \in \mathcal{T}_{q_2}$. Then, there exists a vector $\boldsymbol{u}'$ where $\boldsymbol{v}_1^1+\boldsymbol{u}'$ can be satisfied by one bucket from buckets $1-3$ and $\boldsymbol{v}_1^2+\boldsymbol{u}'$ can be satisfied by one bucket from buckets $4-6$. Thus, the code can satisfy $\boldsymbol{v}_1^1$ and $\boldsymbol{v}_1^2$, that consist the request $\boldsymbol{v}_1$, by one bucket from $1-3$, one bucket from $4-6$, and the last bucket, which satisfies the request $(\boldsymbol{u}',\boldsymbol{u}')$ for each possible $\boldsymbol{u}'$. Then, the remaining two buckets of $1-3$ can satisfy $\boldsymbol{v}_2^1$ and the remaining two buckets of $4-6$ can satisfy $\boldsymbol{v}_2^2$. Similarly, if there exist $2 \leqslant q_1, q_2 \leqslant 4$ where $\boldsymbol{v}_2^1 \in \mathcal{T}_{q_1}$ and $\boldsymbol{v}_2^2 \in \mathcal{T}_{q_2}$, the code can satisfy the requests $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ by disjoint sets.

The last case is when $\{\boldsymbol{v}_1^1,\boldsymbol{v}_2^1\} \subseteq \mathcal{T}_1$ and $\{\boldsymbol{v}_1^2,\boldsymbol{v}_2^2\} \subseteq \mathcal{T}_q$, where $2 \leqslant q \leqslant 4$ (or $\{\boldsymbol{v}_1^1,\boldsymbol{v}_2^1\} \subseteq \mathcal{T}_q$ and $\{\boldsymbol{v}_1^2,\boldsymbol{v}_2^2\} \subseteq \mathcal{T}_q$). In the beginning we satisfy $\boldsymbol{v}_1^1$ by one bucket from $1-3$. Then, take a vector $\boldsymbol{u}''$, such that $\boldsymbol{v}_2^2+\boldsymbol{u}''$ can be satisfied by one bucket, denote it by b_1. The vector $\boldsymbol{v}_2^1+\boldsymbol{u}''$ can be satisfied by the remaining two buckets from $1-3$, denote them by b_2, b_3. Then, the request $R_2=\{\boldsymbol{v}_2^1,\boldsymbol{v}_2^2\}$ can be satisfied by $\{b_1,b_2,b_3,7\}$ (where 7 is the last bucket). Lastly, the request $\boldsymbol{v}_1^2$ can be satisfied by the remaining two buckets from $4-6$. Thus, we can conclude that there exists 2 recovering sets for each possible request, and hence, $FB_{2,2}(8,2) \leqslant 7$. ∎

The result in Theorem 18 can be generalized to different values of s.

Corollary 19: $\log_7(2^{s-1}\cdot(2^s-1)) \leqslant FB_{2,2}(s,2) \leqslant 7\cdot\left\lceil \frac{s}{8} \right\rceil$.

Proof: The upper bound is derived from Theorem 18, and Theorem 3(e). The lower bound is obtained from Theorem 5, where $FB_{2,2}(s,2) \geqslant m$ where m is the smallest positive integer such that $\sum_{i=2}^{m} \binom{m}{i} \cdot \left\{ {i \atop 2} \right\} \cdot \left(\sum_{j=1}^{2} \binom{2}{j}\right)^i \geqslant \binom{2^s}{2}$. It is known that $\left\{ {i \atop 2} \right\} = 2^{i-1}-1$. Thus, $\sum_{i=2}^{m} \binom{m}{i} \cdot (2^{i-1}-1) \cdot 3^i \geqslant 2^{s-1} \cdot (2^s-1)$. For each $i \geqslant 2$, $(2^{i-1}-1)\cdot 3^i \leqslant 6^i$. Hence, it must hold that $\sum_{i=0}^{m} \binom{m}{i} \cdot 6^i \geqslant \sum_{i=2}^{m} \binom{m}{i} \cdot 6^i \geqslant 2^{s-1}\cdot(2^s-1)$. From the equality $\sum_{i=0}^{m} \binom{m}{i} \cdot x^i = (x+1)^m$, we get that $\sum_{i=0}^{m} \binom{m}{i} \cdot 6^i = 7^m \geqslant 2^{s-1} \cdot (2^s-1)$. Thus, $FB_{2,2}(s,2) \geqslant m \geqslant \log_7(2^{s-1}\cdot(2^s-1))$. ∎

According to Corollary 19, we get that for s large enough $\log_7(2^{s-1}\cdot(2^s-1)) = \log_7(2^{s-1}) + \log_7(2^s-1) \approx (s-1)\cdot\log_7(2) + s\cdot\log_7(2) = (2s-1)\cdot\log_7(2) \approx 0.71s \lesssim FB_{2,2}(s,2) \leqslant \left\lceil \frac{7s}{8} \right\rceil$.

In addition, the result in Theorem 18 can be modified to different value of t.

Corollary 20: $6 \leqslant FB_{3,1}(8,2) \leqslant 7$.

Proof: The lower bound is obtained from Theorem 5. The upper bound is verified by Theorem 15(b), where $FB_{3,1}(8,2) \leqslant FB_{2,2}(8,2) \leqslant 7$. ∎

VI. Specific Constructions of Array Codes

In this section we discuss three constructions of array codes.

A. Construction A

We start with a construction given in [19, Th.20], where it was proved in [9, Th.10] that this construction gives a PIR array code for any integer $t \geqslant 2$. We study how it can be used also as batch and functional PIR array codes for $t=2$. First, the construction for the general case is presented.

Construction 21: Let $t \geqslant 2$ be a fixed integer. The number of information bits is $s = t(t+1)$, the number of cells in each bucket (the number of the rows) is t. The number of buckets is $m = m' + m''$, where $m' = \binom{t(t+1)}{t}$, and $m'' = \binom{t(t+1)}{t+1}/t$. In the first m' buckets all the tuples of t bits out of the $t(t+1)$ information bits are stored, which needs $\binom{t(t+1)}{t}$ buckets. In the last m'' buckets we store all possible summations of $t+1$ bits, such that each one of the $t(t+1)$ bits appears in exactly one summation in every bucket (in each summation there are $t+1$ bits and there are t rows). There are $\binom{t(t+1)}{t+1}$ such summations and since there are t rows then t summations can be stored in each bucket, so the number of buckets of this part is $m'' = \binom{t(t+1)}{t+1}/t$.

For any integer $t \geqslant 2$ denote the code that is obtained from Construction 21 by $\mathcal{C}_t^A$. Construction 21 for the case of $t=2$ is demonstrated in Table VII.

Now we want to show that the code $\mathcal{C}_2^A$ is a $(6,15,25,2,2)$ batch array code, by using several properties which are proved in the following three lemmas. For each $i \in [6]$, denote by $\mathcal{F}_i \subseteq [15]$ the subset of buckets from the first 15 buckets, that have a cell with the singleton x_i. It holds that for any $i \in [6]$, $|\mathcal{F}_i| = 5$, and for any different $i,j \in [6]$, $|\mathcal{F}_i \cap \mathcal{F}_j| = 1$. Assume that every multiset request R of size $k=15$ is represented by a vector $(k_1,\ldots,k_6)$, where k_i indicates the number of times x_i appears in the multiset request and $k_1 \geqslant \cdots \geqslant k_6$.

Lemma 22: For any multiset request $(k_1,\ldots,k_6)$ of size $k=15$, the code $\mathcal{C}_2^A$ can satisfy all the requests of bits x_3,x_4,x_5,x_6 by using only the first 15 buckets.

Proof: The proof is divided into the following cases according to number of different information bits that appear in the request.

Case 1: If $k_3 = 0$, then none of the bits x_3,x_4,x_5,x_6 is requested and the property clearly holds.

Case 2: If $k_4 = 0$, then it necessarily holds that $k_3 \leqslant 5$. Assume by contradiction that $k_3 > 5$. Then, it holds that $k_1 \geqslant k_2 > 5$, and hence, $k = k_1+k_2+k_3 > 15$, which is a contradiction. Thus $k_3 \leqslant 5$ and the code can use k_3 buckets from $\mathcal{F}_3$.

Case 3: If $k_5 = 0$, then it necessarily holds that $k_4 \leqslant k_3 \leqslant 4$. Assume by contradiction that $k_4 > 4$. Then, it holds that $k_1 \geqslant k_2 \geqslant k_3 > 4$, and hence, $k = k_1+k_2+k_3+k_4 > 15$, which is a contradiction. Assume by contradiction that $k_3 > 4$, when $k_4 \geqslant 1$. Then, it holds that $k_1 \geqslant k_2 > 4$, and hence, $k = k_1+k_2+k_3+k_4 > 15$, which is a contradiction. Thus $k_3 \leqslant 4$ and the code $\mathcal{C}_2^A$ can satisfy the bit requests of x_3 by taking k_3 buckets from $\mathcal{F}_3$. Then the code $\mathcal{C}_2^A$ can satisfy the bit requests of x_4 by taking $k_4 \leqslant 4$ buckets from $\mathcal{F}_4 \setminus (\mathcal{F}_4 \cap \mathcal{F}_3)$, where $|\mathcal{F}_4 \setminus (\mathcal{F}_4 \cap \mathcal{F}_3)| = 4$.

Case 4: If $k_6 = 0$, then it necessarily holds that $k_5 \leqslant k_4 \leqslant 3$ and $k_3 \leqslant 4$. Assume by contradiction that $k_5 > 3$. Then,

TABLE VII
CONSTRUCTION 21 FOR $t = 2$

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
$x_1x_2x_3$	$x_1x_2x_4$	$x_1x_2x_5$	$x_1x_2x_6$	$x_1x_3x_4$	$x_1x_3x_5$	$x_1x_3x_6$	$x_1x_4x_5$	$x_1x_4x_6$	$x_1x_5x_6$
$x_4x_5x_6$	$x_3x_5x_6$	$x_3x_4x_6$	$x_3x_4x_5$	$x_2x_5x_6$	$x_2x_4x_6$	$x_2x_4x_5$	$x_2x_3x_6$	$x_2x_3x_5$	$x_2x_3x_4$

it holds that $k_1 \geqslant k_2 \geqslant k_3 \geqslant k_4 \geqslant k_5 > 3$, and hence, $k = k_1 + k_2 + k_3 + k_4 + k_5 > 15$, which is a contradiction. Assume by contradiction that $k_4 > 3$, when $k_5 \geqslant 1$. Then, it holds that $k_1 \geqslant k_2 \geqslant k_3 \geqslant k_4 > 3$, and hence, $k = k_1+k_2+k_3+k_4+k_5 > 15$, which is a contradiction. Assume by contradiction that $k_3 > 4$, when $k_5+k_4 \geqslant 2$. Then, it holds that $k_1 \geqslant k_2 \geqslant k_3 > 4$, and hence, $k = k_1 + k_2 + k_3 + k_4 + k_5 > 15$, which is a contradiction. Thus, $k_3 \leqslant 4$ and the code $\mathcal{C}_2^A$ can satisfy the bit requests of x_3 by taking k_3 buckets from $\mathcal{F}_3$. Also, $k_4 \leqslant 3$, then the code $\mathcal{C}_2^A$ can satisfy the bit requests of x_4 by taking k_4 buckets from $\mathcal{F}_4 \setminus (\mathcal{F}_4 \cap \mathcal{F}_3)$. Lastly, the code $\mathcal{C}_2^A$ can satisfy the bit requests of x_5 by taking $k_5 \leqslant 3$ buckets from $\mathcal{F}_5 \setminus ((\mathcal{F}_5 \cap \mathcal{F}_4) \cup (\mathcal{F}_5 \cap \mathcal{F}_3))$, where $|\mathcal{F}_5 \setminus ((\mathcal{F}_5 \cap \mathcal{F}_4) \cup (\mathcal{F}_5 \cap \mathcal{F}_3))| = 3$.

Case 5: If $k_6 > 0$, then it necessarily holds that $k_6 \leqslant k_5 \leqslant 2$, $k_4 \leqslant 3$ and $k_3 \leqslant 4$. Assume by contradiction that $k_6 > 2$. Then, it holds that $k_1 \geqslant k_2 \geqslant k_3 \geqslant k_4 \geqslant k_5 > 2$, and hence, $k = \sum_{i=1}^{6} k_i > 15$, which is a contradiction. Assume by contradiction that $k_5 > 2$ when $k_6 \geqslant 1$. Then, it holds that $k_1 \geqslant k_2 \geqslant k_3 \geqslant k_4 > 2$, and hence, $k = \sum_{i=1}^{6} k_i > 15$, which is a contradiction. Assume by contradiction that $k_4 > 3$ when $k_6 + k_5 \geqslant 2$. Then, it holds that $k_1 \geqslant k_2 \geqslant k_3 > 3$, and hence, $k = \sum_{i=1}^{6} k_i > 15$, which is a contradiction. Assume by contradiction that $k_3 > 4$ when $k_6 + k_5 + k_4 \geqslant 3$. Then, it holds that $k_1 \geqslant k_2 > 4$, and hence, $k = \sum_{i=1}^{6} k_i > 15$, which is a contradiction. Thus, $1 \leqslant k_3 \leqslant 4$ and the code $\mathcal{C}_2^A$ can satisfy the bit requests of x_3 by taking k_3 buckets from $\mathcal{F}_3$. Then the code $\mathcal{C}_2^A$ can satisfy the bit requests of x_4 by taking $k_4 \leqslant 3$ buckets from $\mathcal{F}_4 \setminus (\mathcal{F}_4 \cap \mathcal{F}_3)$. Then the code $\mathcal{C}_2^A$ can satisfy the bit requests of x_5 by taking $k_5 \leqslant 2$ buckets from $\mathcal{F}_5 \setminus ((\mathcal{F}_5 \cap \mathcal{F}_4) \cup (\mathcal{F}_5 \cap \mathcal{F}_3))$. Lastly, the code $\mathcal{C}_2^A$ can satisfy the bit requests of x_6 by taking $k_6 \leqslant 2$ buckets from $\mathcal{F}_6 \setminus ((\mathcal{F}_6 \cap \mathcal{F}_5) \cup (\mathcal{F}_6 \cap \mathcal{F}_4) \cup (\mathcal{F}_6 \cap \mathcal{F}_3))$, where $|\mathcal{F}_6 \setminus ((\mathcal{F}_6 \cap \mathcal{F}_5) \cup (\mathcal{F}_6 \cap \mathcal{F}_4) \cup (\mathcal{F}_6 \cap \mathcal{F}_3))| = 2$. ∎

Lemma 23: In the code $\mathcal{C}_2^A$, for any information bit x_i and for any bucket $b_1 \in [15] \setminus \mathcal{F}_i$, there exists a bucket $b_2, 16 \leqslant b_2 \leqslant 25$ such that $\{b_1, b_2\}$ is a recovering set of x_i. In addition, the $|[15] \setminus \mathcal{F}_i|$ recovering sets are mutually disjoint.

Proof: For any information bit x_i, the buckets of $[15] \setminus \mathcal{F}_i$, are the buckets from the first $m' = 15$ buckets that does not include x_i. Each bucket $b_1 \in [15] \setminus \mathcal{F}_i$ has two singletons x_{j_1}, x_{j_2} which are different than x_i. From the construction of the code $\mathcal{C}_2^A$ we know that there exists a bucket b_2 from the last 10 buckets that has the summation $x_i + x_{j_1} + x_{j_2}$. Thus, the subset $\{b_1, b_2\}$ is a recovering set of x_i.

We want to show that for any two different buckets $b_1', b_1'' \in [15] \setminus \mathcal{F}_i$, the recovering sets $\{b_1', b_2'\}$ and $\{b_1'', b_2''\}$ of x_i are disjoint. Assume that b_1' has the two bits $x_{j_1'}, x_{j_2'}$ and b_1'' has the two bits $x_{j_1''}, x_{j_2''}$. Assume that $x_{j_1'} \neq x_{j_1''}$. It holds that $\{b_1'\} \cap \{b_1'', b_2''\} = \emptyset$ because it holds that $b_1' \neq b_1''$ and $b_1' \neq b_2''$ because $b_1' \in [15]$ but $b_2'' \notin [15]$. In addition, $\{b_2'\} \cap \{b_1'', b_2''\} = \emptyset$ because it holds that $b_2' \notin [15]$ but $b_1'' \in [15]$. Also, $b_2' \neq b_2''$ because b_2' has the summation $x_i + x_{j_1'} + x_{j_2'}$ and b_2'' has the summation $x_i + x_{j_1''} + x_{j_2''}$. The two summations are different because $x_{j_1'} \neq x_{j_1''}$. From the construction we know that each bucket in the last 10 buckets has exactly one summation with x_i. Thus, $b_2' \neq b_2''$. ∎

For any information bit $x_i, i \in [6]$ denote by R_b^i the recovering set that uses bucket $b \in [15]$ and can satisfy x_i. For example, $R_1^1 = \{1\}$ and $R_{12}^1 = \{12, 22\}$.

Lemma 24: For the two information bits x_1, x_2, the buckets $\{10, 11, \ldots, 15\}$ are divided into 3 pairs, $\mathcal{P} = \{(10, 15), (11, 14), (12, 13)\}$, such that for any pair $(b_1, b_2) \in \mathcal{P}$, it holds that $\left|R_{b_1}^1 \cap R_{b_2}^2\right| > 0$ and $\left|R_{b_1}^2 \cap R_{b_2}^1\right| > 0$.

Proof: For the first pair, $(10, 15)$, it holds that $R_{10}^1 = \{10, 20\}, R_{10}^2 = \{10, 25\}, R_{15}^1 = \{15, 25\}$, and $R_{15}^2 = \{15, 20\}$. Then, it holds that $\left|R_{10}^1 \cap R_{15}^2\right| = |\{10, 20\} \cap \{15, 20\}| > 0$ and $\left|R_{10}^2 \cap R_{b_{15}}^1\right| = |\{10, 25\} \cap \{15, 25\}| > 0$. Similarly, the claim holds also for the pairs $(11, 14)$ and $(12, 13)$. ∎

Now, we are ready to show that the code $\mathcal{C}_2^A$ is a $(6, 15, 25, 2, 2)$ batch array code.

Theorem 25: The code $\mathcal{C}_2^A$ is a $(6, 15, 25, 2, 2)$ batch array code. In particular, $B_{2,2}(6, 15) = 25$.

Proof: The lower bound is derived from Theorem 2(c), $B_{2,2}(6, 15) \geqslant \frac{30 \cdot 6 \cdot 7}{(4)^2 + 36 - 4 + 4} > 24$. The upper bound is derived from the code $\mathcal{C}_2^A$. Let $(k_1, \ldots, k_6)$ be a multiset request of size $k = 15$. The first step is to satisfy all the requests of bits x_3, x_4, x_5, x_6 according to Lemma 22 by using only the first $m' = 15$ buckets. Then, the remaining requests are of the bits x_1, x_2. Denote by α_1, α_2 the number of the remaining buckets from the first $m' = 15$ buckets that include x_1, x_2 as singleton, but not both of them, respectively. Then, take $\min\{k_2, \alpha_2\}$ buckets as a recovering sets of x_2 and take $\min\{k_1, \alpha_1\}$ buckets as recovering sets of x_1. The first bucket which contains the singletons x_1, x_2 is not used yet. Denote by r the number of bit requests from the multiset request that were satisfied so far. Furthermore, denote by k_1', k_2' the number of remaining bit requests of x_1, x_2, respectively, where $k_1' = k_1 - \min\{k_1, \alpha_1\}$ and $k_2' = k_2 - \min\{k_2, \alpha_2\}$. After this step we still have $15 - r$ buckets in the first $m' = 15$ buckets, including the first bucket and all the last $m'' = 10$ buckets. Therefore, for x_1 and x_2 there are $15 - r$ possible recovering sets.

The second step is to satisfy the remaining $15 - r$ bit requests from the multiset request. If $k_1' = 0$ or $k_2' = 0$, then it is possible to satisfy them by using the remaining $k - r = 15 - r$ recovering sets of x_1 or x_2. Otherwise, $k_1' > 0$ and $k_2' > 0$. So far we used all the buckets from the set $(\mathcal{F}_1 \cup \mathcal{F}_2) \setminus \{1\}$ which is of size 8 and another p buckets from the subset $\{10, 11, \ldots, 15\}$. Thus, $k_1' + k_2' = 7 - p$. Let $\mathcal{G} \subseteq \{10, 11, \ldots, 15\}$ be the subset of buckets from $\{10, 11, \ldots, 15\}$ that were not used in the first step and let $p = 6 - |\mathcal{G}|$. According to Lemma 23, there are at least $7 - p$ remaining recovering sets for each bit of $\{x_1, x_2\}$, which are the set $\{1\}$ and the sets of R_b^i where $b \in \mathcal{G}$ and $i \in$ [2]. According to Lemma 24, the buckets $\{10, 11, \ldots, 15\}$ are divided into 3 pairs, where the b-th bucket is paired with the $(25 - b)$-th bucket, for $10 \leqslant b \leqslant 15$. The subset $\mathcal{G}$ is partitioned into two subsets, $\mathcal{U}_1 = \{b \in \mathcal{G} : (25 - b) \in \mathcal{G}\}$ and $\mathcal{U}_2 = \{b \in \mathcal{G} : (25 - b) \notin \mathcal{G}\}$. Let $\beta_1 = |\mathcal{U}_1|$ and $\beta_2 = |\mathcal{U}_2|$. The following cases are considered.

Case 1: If p is even and k_1' is even (or k_2' is even). Since p is even, it is deduced that β_2 is even as well. Assume that k_1' is even, then also $(k_1' - \beta_2)$ is even. In order to satisfy x_1 we can take $\min\{\beta_2, k_1'\}$ recovering sets that use $\min\{\beta_2, k_1'\}$ buckets from $\mathcal{U}_2$. We can see that $\beta_1 + \beta_2 = 6 - p$ and $k_1' \leqslant 6 - p = \beta_1 + \beta_2$ then $k_1' - \beta_2 \leqslant \beta_1$. If $k_1' > \beta_2$, then we can satisfy the remaining requests of x_1 with $(k_1' - \beta_2)/2$ pairs of buckets from $\mathcal{U}_1$, where for each bucket b from the $(k_1' - \beta_2)$ buckets we can take R_b^1 as a recovering set for x_1. It is possible to show that each recovering set for x_1 that uses a bucket from $\mathcal{U}_2$ intersects with only one recovering set for x_2 that uses a bucket from $\mathcal{G}$. Also, each pair of recovering sets for x_1 that uses a pair of bucket from $\mathcal{U}_1$ intersects with only two recovering sets for x_2 that use buckets from $\mathcal{G}$. Thus, from the $7 - p$ recovering sets of x_2 it is not possible to use only $\max\{k_1', \beta_2 + 2 \cdot \frac{k_1' - \beta_2}{2}\} = k_1'$ of them. Thus it is possible to use the remaining $7 - p - k_1' = k_2'$ to satisfy the k_2' requests of x_2. The case when k_1' is odd but k_2' is even can be solved similarly while changing between x_1 and x_2.

Case 2: If p is odd and k_1' is odd (or k_2' is odd). Then β_2 is odd. Assume that k_1' is odd, then also $(k_1' - \beta_2)$ is even and the rest is similar to Case 1.

Case 3: If p is even and k_1', k_2' are odd. Then start with satisfying x_1 with a recovering set $\{1\}$. Then we still have an even number of remaining requests of x_1 that must be satisfied, and the rest is similar to Case 1.

Case 4: If p is odd and k_1', k_2' are even. Then start with satisfying x_1 with a recovering set $\{1\}$. Then we still have an odd number of remaining requests of x_1 that must be satisfied, and the rest is similar to Case 2.

Thus, we can conclude that the code can satisfy each multiset of 15 information bits, and hence, $B_{2,2}(6, 15) = 25$. ∎

In addition it is possible to show that the code $\mathcal{C}_2^A$ is a $(6, 11, 25, 2, 2)$ functional PIR array code.

Theorem 26: The code $\mathcal{C}_2^A$ is a $(6, 11, 25, 2, 2)$ functional PIR array code. In particular, $21 \leqslant FP_{2,2}(6, 11) \leqslant 25$.

Proof: The lower bound is obtained from Theorem 9, where $FP_{2,2}(6, 11) \geqslant \frac{2 \cdot 11 \cdot 63}{3 + 63} = 21$. The upper bound can be obtained from the code $\mathcal{C}_2^A$. Given a request R, a linear combination of the information bits, that the code $\mathcal{C}_2^A$ must satisfy $k = 11$ times by disjoint recovering sets. Because of the symmetry of $x_i, i \in [6]$, it is sufficient to check requests according to their length (number of information bits). Thus, the proof is divided into the following cases according to number of information bits that appear in the request.

Case 1: If the request contains one information bit then it is the case of PIR.

Case 2: If the request contains two information bits, then assume that it is $x_1 + x_2$. Then the recovering sets are the following $\{\{1\}, \{2, 6\}, \{3, 7\}, \{4, 8\}, \{5, 9\}, \{16, 11\}, \{17, 10\}, \{18, 13\}, \{19, 12\}, \{20, 25\}, \{21, 24\}, \{22, 23\}\}$.

Case 3: If the request contains three information bits, then assume that it is $x_1 + x_2 + x_3$. Then the recovering sets are the following $\{\{16\}, \{1, 2\}, \{17, 10\}, \{18, 11\}, \{19, 12\}, \{20, 7\}, \{21, 8\}, \{22, 9\}, \{23, 5\}, \{24, 4\}, \{25, 3\}\}$.

Case 4: If the request contains four information bits, then assume that it is $x_3 + x_4 + x_5 + x_6$. Then the recovering sets are the following $\{\{16, 2\}, \{17, 3\}, \{18, 4\}, \{19, 5\}, \{20, 25\}, \{21, 24\}, \{22, 23\}, \{10, 15\}, \{11, 14\}, \{12, 13\}, \{6, 7, 8, 9\}\}$.

Case 5: If the request contains five information bits, then assume that it is $x_2 + x_3 + x_4 + x_5 + x_6$. Then the recovering sets are the following $\{\{16, 1\}, \{17, 2\}, \{18, 3\}, \{19, 4\}, \{20, 5\}, \{21, 11\}, \{22, 12\}, \{23, 13\}, \{24, 14\}, \{25, 15\}, \{6, 7, 8, 9\}\}$.

Case 6: If the request contains all the information bits, that it is $x_1 + x_2 + x_3 + x_4 + x_5 + x_6$. Then the recovering sets are the following $\{\{16\}, \{17\}, \{18\}, \{19\}, \{20\}, \{21\}, \{22\}, \{23\}, \{24\}, \{25\}, \{1, 10, 15\}, \{2, 8, 14\}, \{3, 9, 11\}, \{4, 7, 12\}, \{5, 6, 13\}\}$. ∎

B. Construction B

Next we generalize an example given in [19] of a PIR code for any integer $r \geqslant 3$ and study how it can be used also as batch array codes. We first present the construction for the general case.

Construction 27: Let $r \geqslant 3$ be a fixed integer, the number of information bits is $s = r(r + 1)$, the number of the buckets is $m = r + 1$, and the number of the cells in each bucket is $t = (r - 1)r + 1$. The information bits are partitioned into $r + 1$ parts each of size r, denote by $\mathcal{S}_i$ the part i of the bits. For each $i \in [r + 1]$, write the linear combination $\sum_{j \in \mathcal{S}_i} x_j$ to bucket i. For each $i, i \in [r + 1]$ write each one of the subsets of size $r - 1$ of $\mathcal{S}_i$ as singletons in a different bucket other than bucket i.

For any integer $r \geqslant 3$ denote the code that is obtained from Construction 27 by $\mathcal{C}_r^B$. Construction 27 for the case of $r = 3$ is demonstrated in Table VIII. It is possible to show that for any $r \geqslant 3$ the code $\mathcal{C}_r^B$ is an $(r^2 + r, r, r + 1, r^2 - r + 1, r - 1)$ PIR array code.

Theorem 28: For any integer $r \geqslant 3$ the code $\mathcal{C}_r^B$ from Construction 27 is an $(r^2 + r, r, r + 1, r^2 - r + 1, r - 1)$ PIR array code. In particular,

$$\frac{r \cdot (4r^2 + 3r - 1)}{4r^2 - r + 1} \leqslant P_{r^2 - r + 1, r - 1}(r^2 + r, r) \leqslant r + 1.$$

TABLE VIII
CONSTRUCTION 27 FOR $r = 3$

1	2	3	4
$x_1x_2x_3$	x_1	x_2	x_1
x_4	x_2	x_3	x_3
x_6	$x_4x_5x_6$	x_4	x_5
x_7	x_7	x_5	x_6
x_8	x_9	$x_7x_8x_9$	x_8
x_{10}	x_{10}	x_{11}	x_9
x_{11}	x_{12}	x_{12}	$x_{10}x_{11}x_{12}$

Proof: The lower bound can be obtained by using Theorem 2(b),

$$\begin{aligned} P_{r^2-r+1,r-1}(r^2+r,r) &\geqslant P_{r^2-r+1,r^2-r+1}(r^2+r,r) \\ &\geqslant \frac{r\cdot(r^2+r)(4r-1)}{(4r-1)(r^2-r+1)+(2r-1)^2} \\ &= \frac{r(4r^3-r^2+4r^2-r)}{4r^3-4r^2+4r-r^2+r-1+4r^2-4r+1} \\ &= \frac{r^2(4r^2+3r-1)}{4r^3-r^2+r} = \frac{r\cdot(4r^2+3r-1)}{4r^2-r+1}. \end{aligned}$$

The upper bound is verified by using the code $\mathcal{C}_r^B$. There are $s = r(r+1)$ information bits, and the number of buckets is $m = r+1$. For each $i \in [m]$, there exists a cell with the linear combination $\sum_{q\in\mathcal{S}_i} x_q$ and another $r(r-1)$ cells to store one $(r-1)$-subset from each $\mathcal{S}_j, j \in [r+1]$, where $j \neq i$. Thus, the number of the rows is r^2-r+1.

Let x_j be a request that the code $\mathcal{C}_r^B$ must satisfy by r disjoint recovering sets. Assume that $x_j \in \mathcal{S}_i, i \in [r+1]$. There are $r-1$ buckets which include x_j as a singleton, because x_j appears in $r-1$ subsets of length $r-1$ of part $\mathcal{S}_i$. Thus, each bucket of the $r-1$ buckets is taken as a recovering set, while reading only one cell from it. In addition, in the i-th bucket there exists a cell with $\sum_{q\in\mathcal{S}_i} x_q$, which includes x_j. The $(r-1)$-subset, $\mathcal{S}_i \setminus \{x_j\}$, is written in a bucket p, which is different from bucket i, and is different from the buckets that were taken so far (because $x_j \notin \mathcal{S}_i \setminus \{x_j\}$). Thus, the set $\{i,p\}$ is a recovering set of x_j, and it is sufficient to read from bucket i one cell, which is $\sum_{q\in\mathcal{S}_i} x_q$ and to read $r-1$ cells with the $r-1$ bits of $\mathcal{S}_i \setminus \{x_j\}$ from bucket p. Thus, there exist r disjoint recovering sets for x_j, where at most $r-1$ cells are read from each bucket. ∎

Next we want to show that for any integer $r \geqslant 3$ the code $\mathcal{C}_r^B$ is an $(r^2+r, r, r+1, r^2-r+1, r-1)$ batch array code, by using a property stated in the following lemma.

Lemma 29: For any integer $r \geqslant 3$ it holds that every two buckets of the code $\mathcal{C}_r^B$ can form a recovering set of every bit x_i by reading at most $r-1$ cells from each bucket.

Proof: Given a pair of buckets from $\mathcal{C}_r^B$, for simplicity we assume that they are the first two buckets. The first bucket has a cell with $\sum_{i\in\mathcal{S}_1} x_i$, and has exactly $r-1$ bits as singletons from each $\mathcal{S}_j, 2 \leqslant j \leqslant r+1$. Hence, the first bucket does not include exactly one of the information bits from each $\mathcal{S}_j, 2 \leqslant j \leqslant r+1$. Thus, the number of bits that do not appear as singletons in the first bucket is $2r$. Hence, the first bucket can satisfy each information bit except to these $2r$ bits, by reading exactly one cell.

The second bucket contains $r-1$ bits out of the r bits of $\mathcal{S}_1$ as singletons. Thus, each one of these $(r-1)$ bits from $\mathcal{S}_1$ can be satisfied by reading each one of them as a singleton from the second bucket. Also, the remaining bit of $\mathcal{S}_1$ can be satisfied by reading the $r-1$ singletons of $\mathcal{S}_1$ from the second bucket with the cell $\sum_{i\in\mathcal{S}_1} x_i$ in the first bucket.

The first two buckets include different $(r-1)$-subsets of each part other than $\mathcal{S}_1, \mathcal{S}_2$. Then, the information bit that does not appear as a singleton cell or as part of the cell $\sum_{i\in\mathcal{S}_1} x_i$ in the first bucket, definitely appears as a singleton cell or in the cell $\sum_{i\in\mathcal{S}_2} x_i$ in the second bucket. Then, each bit $x_q \in \mathcal{S}_j$ where $3 \leqslant j \leqslant r+1$ can be satisfied by reading it as a singleton from the second bucket. There are $r-1$ such bits, and thus, it remains to show that the code can satisfy the bit $x_{q_1} \in \mathcal{S}_2$ that is not part of the $(r-1)$-subset of singletons which are stored in the first bucket. We can satisfy x_{q_1} by reading the $r-1$ singletons of $\mathcal{S}_2$ from the first bucket with the cell $\sum_{i\in\mathcal{S}_2} x_i$ in the second bucket. Thus, the first two buckets of the code $\mathcal{C}_r^B$ can form a recovering set of every bit x_i. Similarly, it holds for any two buckets of the code $\mathcal{C}_r^B$. ∎

Now, we are ready to show that for any integer $r \geqslant 3$ the code $\mathcal{C}_r^B$ is $(r^2+r, r, r+1, r^2-r+1, r-1)$ batch array code.

Theorem 30: For any integer $r \geqslant 3$ the code $\mathcal{C}_r^B$ from Construction 27 is an $(r^2+r, r, r+1, r^2-r+1, r-1)$ batch array code. In particular,

$$\frac{r\cdot(4r^2+3r-1)}{4r^2-r+1} \leqslant B_{r^2-r+1,r-1}(r^2+r,r) \leqslant r+1.$$

Proof: The lower bound is follows from the lower bound of $P_{r^2-r+1,r-1}(r^2+r,r)$. The upper bound is achieved by using Contruction 27. Let $R = \{x_{i_1}, x_{i_2}, \ldots, x_{i_r}\}$ be a multiset request of r information bits. First, we want to show that the code $\mathcal{C}_r^B$ can satisfy the first $r-1$ bits of the request by using only $r-1$ buckets. From Construction 27 it is known that each information bit x_i appears as a singleton in $r-1$ buckets out of the $r+1$ buckets. Thus, in each subset of buckets of size at least 3, there is at least one bucket that contains a cell with x_i. Therefore, the first $r-1$ bits of the request can be read by singletons from $r-1$ different buckets.

After the first step, we still have 2 buckets and from Lemma 29 it is known that these two buckets can satisfy each x_i, in particular x_{i_r}. ∎

According to Theorem 28 and Theorem 30 it can be verified that for any $r \geqslant 3$, $r < \frac{r\cdot(4r^2+3r-1)}{4r^2-r+1} \leqslant P_{r^2-r+1,r-1}(r^2+r,r) \leqslant B_{r^2-r+1,r-1}(r^2+r,r) \leqslant r+1$. Thus, we conclude that Construction 27 gives optimal PIR and batch array codes.

C. Construction C

We now present our third construction, and study how it can be used as PIR and functional PIR array codes for specific parameters.

Construction 31: Let $s \geqslant 2$ be a fixed integer. The number of information bits is s and the number of the cells in each bucket (the number of the rows) is 2. Let $\mathcal{G}$ be the set of all linear combinations of the s information bits. All the different disjoint pairs from the set $\mathcal{G}$ are stored where each disjoint

TABLE IX
CONSTRUCTION 31 FOR $s = 4$

1	2	3	4	5	6	7	8	9	10	11	12	13	14
x_1	x_1	x_1	x_2	x_2	x_3	x_1	x_1	x_1	x_2	x_2	x_2	x_3	x_3
x_2	x_3	x_4	x_3	x_4	x_4	x_2x_3	x_2x_4	x_3x_4	x_1x_3	x_1x_4	x_3x_4	x_1x_2	x_1x_4

15	16	17	18	19	20	21	22	23	24	25
x_3	x_4	x_4	x_4	x_1	x_2	x_3	x_4	x_1x_2	x_1x_3	x_1x_4
x_2x_4	x_1x_2	x_1x_3	x_2x_3	$x_2x_3x_4$	$x_1x_3x_4$	$x_1x_2x_4$	$x_1x_2x_3$	x_3x_4	x_2x_4	x_2x_3

pair is stored in one bucket. Each disjoint pair has at most s information bits. Thus, we need $m = \sum_{i=2}^{s}(\binom{s}{i} \cdot \left\{ {i \atop 2} \right\})$ buckets. Then,

$$m = \sum_{i=2}^{s}\left(\binom{s}{i}\left\{ {i \atop 2} \right\}\right) = \sum_{i=2}^{s}\binom{s}{i}(2^{i-1}-1) = \frac{3^s+1}{2} - 2^s.$$

For any integer $s \geqslant 2$ denote the code that is obtained from Construction 31 by $\mathcal{C}_s^C$. Construction 31 for the case of $s = 4$ is demonstrated in Table IX and provides the following results. First, we show that the code $\mathcal{C}_4^C$ is a $(4, 16, 25, 2, 1)$ PIR array code.

Theorem 32: The code $\mathcal{C}_4^C$ from Construction 31 is a $(4, 16, 25, 2, 1)$ PIR array code. In particular, $23 \leqslant P_{2,1}(4, 16) \leqslant 25$.

Proof: The lower bound is obtained using Theorem 2(b), $P_{2,1}(4, 16) \geqslant P_{2,2}(4, 16) \geqslant \frac{16\cdot 4\cdot 5}{5\cdot 2+4} > 22$. The upper bound is verified using the code $\mathcal{C}_4^C$. Let $x_i, i \in [4]$ be a request, that the code $\mathcal{C}_4^C$ must satisfy 16 times. From the symmetry of the code, assume that $x_i = x_1$. The following are the recovering sets of x_1, where from each bucket only one cell is read. $\{\{1\},\{2\}, \{3\}, \{7\}, \{8\}, \{9\}, \{19\}, \{10, 6\}, \{11, 5\}, \{13, 4\}, \{14, 18\}, \{15, 17\}, \{16, 12\}, \{20, 23\}, \{21, 24\}, \{22, 25\}\}$. ■

Next, we show that the code $\mathcal{C}_4^C$ is a $(4, 14, 25, 2, 2)$ functional PIR array code.

Theorem 33: The code $\mathcal{C}_4^C$ from Construction 31 is a $(4, 14, 25, 2, 2)$ functional PIR array code. In particular, $24 \leqslant FP_{2,2}(4, 14) \leqslant 25$.

Proof: The lower bound is obtained using Theorem 9, $FP_{2,2}(4, 14) \geqslant \frac{2\cdot 14\cdot 15}{15+3} > 23$. The upper bound is verified using the code $\mathcal{C}_4^C$. Let R be a linear combination request, that the code $\mathcal{C}_4^C$ must satisfy 14 times. From the symmetry of the code, the proof is divided into the following cases according to the number of information bits that appear in R. If the number of information bits that appear in R is p then we assume that the request is $x_1 + x_2 + \cdots + x_p$.

Case 1: The recovering sets are the following $\{\{1\}, \{2\}, \{3\}, \{7\}, \{8\}, \{9\}, \{19\}, \{10, 6\}, \{11, 5\}, \{13, 4\}, \{14, 18\}, \{15, 17\}, \{16, 12\}, \{20, 23\}, \{21, 24\}, \{22, 25\}\}$.

Case 2: The recovering sets are the following $\{\{1\}, \{13\}, \{16\}, \{23\}, \{2, 4\}, \{3, 5\}, \{7, 10\}, \{8, 11\}, \{9, 12\}, \{14, 15\}, \{17, 18\}, \{19, 20\}, \{21, 22\}, \{24, 25\}\}$.

Case 3: The recovering sets are the following. $\{\{7\}, \{10\}, \{13\}, \{22\}, \{1, 24\}, \{2, 23\}, \{3, 25\}, \{4, 17\}, \{5, 14\}, \{6, 16\}, \{8, 20\}, \{9, 21\}, \{11, 12\}, \{18, 19\}\}$.

Case 4: The recovering sets are the following. $\{\{19\}, \{20\}, \{21\}, \{22\}, \{23\}, \{24\}, \{25\}, \{1, 6\}, \{2, 5\}, \{3, 4\}, \{7, 11\}, \{8, 10\}, \{9, 13\}, \{12, 16\}, \{14, 18\}, \{15, 17\}\}$. ■

Construction 31 for the case of $s = 5$ is demonstrated in Table X and provides the following result.

Theorem 34: The code $\mathcal{C}_5^C$ from Construction 31 is a $(5, 48, 90, 2, 2)$ functional PIR array code. In particular, $88 \leqslant FP_{2,2}(5, 48) \leqslant 90$.

Proof: The lower bound is obtained using Theorem 9, $FP_{2,2}(5, 48) \geqslant \frac{2\cdot 48\cdot 31}{31+3} > 87$. The upper bound is verified using the code $\mathcal{C}_5^C$. Let R be a linear combination request that the code $\mathcal{C}_5^C$ must satisfy 48 times. From the symmetry of the code, the proof is divided into the following cases according to the number of information bits that appear in R. If the number of information bits that appear in R is p then we assume that the request is $x_1 + x_2 + \cdots + x_p$.

Case 1: The recovering sets are the following $\{\{1\}, \{2\}, \{3\}, \{4\}, \{11\}, \{12\}, \{13\}, \{14\}, \{15\}, \{16\}, \{41\}, \{42\}, \{43\}, \{44\}, \{61\}, \{17, 26\}, \{18, 32\}, \{19, 38\}, \{20, 23\}, \{21, 29\}, \{22, 35\}, \{24, 33\}, \{25, 39\}, \{27, 30\}, \{28, 36\}, \{31, 40\}, \{34, 37\}, \{45, 8\}, \{46, 9\}, \{47, 10\}, \{49, 6\}, \{50, 7\}, \{48, 51\}, \{53, 5\}, \{52, 54\}, \{55, 67\}, \{57, 72\}, \{58, 69\}, \{59, 66\}, \{64, 56\}, \{65, 60\}, \{62, 78\}, \{63, 79\}, \{71, 81\}, \{73, 82\}, \{83, 80\}, \{68, 85\}, \{70, 88\}, \{74, 86\}, \{75, 90\}, \{76, 89\}, \{77, 87\}\}$.

Case 2: The recovering sets are the following $\{\{1\}, \{23\},\{29\}, \{35\}, \{66\}, \{67\}, \{68\}, \{81\}, \{2, 5\}, \{3, 6\}, \{4, 7\}, \{9, 53\}, \{8, 49\}, \{10, 88\}, \{11, 20\}, \{12, 21\}, \{13, 22\}, \{14, 17\}, \{15, 18\}, \{16, 19\}, \{24, 26\}, \{25, 27\}, \{30, 32\}, \{31, 33\}, \{36, 38\}, \{37, 39\}, \{41, 45\}, \{42, 46\}, \{43, 47\}, \{44, 85\}, \{51, 52\}, \{54, 57\}, \{55, 56\}, \{59, 60\}, \{61, 28\}, \{62, 34\}, \{63, 40\}, \{64, 74\}, \{65, 77\}, \{69, 80\}, \{70, 79\}, \{71, 72\}, \{73, 78\}, \{75, 82\}, \{84, 87\}, \{86, 48\}, \{50, 58\}, \{76, 83\}\}$.

Case 3: The recovering sets are the following $\{\{11\}, \{17\}, \{23\}, \{53\}, \{57\}, \{90\}, \{1, 8\}, \{2, 6\}, \{3, 32\}, \{4, 38\}, \{5, 16\}, \{7, 36\}, \{9, 29\}, \{10, 89\}, \{30, 66\}, \{12, 40\}, \{13, 34\}, \{14, 39\}, \{15, 33\}, \{18, 80\}, \{19, 79\}, \{20, 37\}, \{21, 31\}, \{22, 88\}, \{24, 76\}, \{73, 86\}, \{26, 74\}, \{27, 68\}, \{28, 87\}, \{35, 63\}, \{41, 64\}, \{42, 65\}, \{43, 81\}, \{44, 47\}, \{45, 69\}, \{46, 60\}, \{48, 82\}, \{49, 56\}, \{50, 59\}, \{51, 78\}, \{52, 85\}, \{54, 67\}, \{55, 70\}, \{58, 77\}, \{61, 75\}, \{62, 71\}, \{72, 84\}, \{25, 83\}\}$.

Case 4: The recovering sets are the following $\{\{41\}, \{45\}, \{49\}, \{53\}, \{65\}, \{66\}, \{69\}, \{72\}, \{1, 8\}, \{2, 6\}, \{3, 5\}, \{10, 11\}, \{9, 12\}, \{7, 14\}, \{4, 20\}, \{13, 28\}, \{15, 22\}, \{16, 21\}, \{17, 34\}, \{18, 27\}, \{19, 40\}, \{23, 64\}, \{24, 62\}, \{25, 33\}, \{26, 61\}, \{29, 63\}, \{30, 48\}, \{31, 38\}, \{32, 44\}, \{35, 88\}, \{36, 39\}, \{37, 85\}, \{42, 90\}, \{43, 89\}, \{46, 68\}, \{47, 67\}, \{50, 71\}, \{51, 87\}, \{52, 84\}, \{54, 74\}, \{55, 70\}, \{56, 75\}, \{57, 78\}, \{58, 79\}, \{59, 73\}, \{60, 76\}, \{77, 81\}, \{82, 86\}\}$.

TABLE X
CONSTRUCTION 31 FOR $s = 5$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
x_1	x_1	x_1	x_1	x_2	x_2	x_2	x_3	x_3	x_4	x_1	x_1	x_1	x_1	x_1	x_1	x_2	x_2	x_2	x_2	x_2	x_2
x_2	x_3	x_4	x_5	x_3	x_4	x_5	x_4	x_5	x_5	x_2x_3	x_2x_4	x_2x_5	x_3x_4	x_3x_5	x_4x_5	x_1x_3	x_1x_4	x_1x_5	x_3x_4	x_3x_5	x_4x_5

23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
x_3	x_3	x_3	x_3	x_3	x_3	x_4	x_4	x_4	x_4	x_4	x_4	x_5	x_5	x_5	x_5	x_5	x_5
x_1x_2	x_1x_4	x_1x_5	x_2x_4	x_2x_5	x_4x_5	x_1x_2	x_1x_3	x_1x_5	x_2x_3	x_2x_5	x_3x_5	x_1x_2	x_1x_3	x_1x_4	x_2x_3	x_2x_4	x_3x_4

41	42	43	44	45	46	47	48	49	50	51	52	53	54
x_1	x_1	x_1	x_1	x_2	x_2	x_2	x_2	x_3	x_3	x_3	x_3	x_4	x_4
$x_2x_3x_4$	$x_2x_3x_5$	$x_2x_4x_5$	$x_3x_4x_5$	$x_1x_3x_4$	$x_1x_3x_5$	$x_1x_4x_5$	$x_3x_4x_5$	$x_1x_2x_4$	$x_1x_2x_5$	$x_1x_4x_5$	$x_2x_4x_5$	$x_1x_2x_3$	$x_1x_2x_5$

55	56	57	58	59	60	61	62	63	64	65
x_4	x_4	x_5	x_5	x_5	x_5	x_1	x_2	x_3	x_4	x_5
$x_1x_3x_5$	$x_2x_3x_5$	$x_1x_2x_3$	$x_1x_2x_4$	$x_1x_3x_4$	$x_2x_3x_4$	$x_2x_3x_4x_5$	$x_1x_3x_4x_5$	$x_1x_2x_4x_5$	$x_1x_2x_3x_5$	$x_1x_2x_3x_4$

66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
x_1x_2	x_1x_2	x_1x_2	x_1x_3	x_1x_3	x_1x_3	x_1x_4	x_1x_4	x_1x_4	x_1x_5	x_1x_5	x_1x_5	x_2x_3	x_2x_4	x_2x_5
x_3x_4	x_3x_5	x_4x_5	x_2x_4	x_2x_5	x_4x_5	x_2x_3	x_2x_5	x_3x_5	x_2x_3	x_2x_4	x_3x_4	x_4x_5	x_3x_5	x_3x_4

81	82	83	84	85	86	87	88	89	90
x_1x_2	x_1x_3	x_1x_4	x_1x_5	x_2x_3	x_2x_4	x_2x_5	x_3x_4	x_3x_5	x_4x_5
$x_3x_4x_5$	$x_2x_4x_5$	$x_2x_3x_5$	$x_2x_3x_4$	$x_1x_4x_5$	$x_1x_3x_5$	$x_1x_3x_4$	$x_1x_2x_5$	$x_1x_2x_4$	$x_1x_2x_3$

Case 5: The recovering sets are the following $\{\{61\}, \{62\}, \{63\}, \{64\}, \{65\}, \{81\}, \{82\}, \{83\}, \{84\}, \{85\}, \{86\}, \{87\}, \{88\}, \{89\}, \{90\}, \{66, 4\}, \{67, 3\}, \{68, 2\}, \{69, 7\}, \{70, 6\}, \{71, 1\}, \{72, 9\}, \{73, 5\}, \{74, 17\}, \{75, 10\}, \{76, 8\}, \{77, 18\}, \{78, 11\}, \{79, 12\}, \{80, 13\}, \{41, 40\}, \{42, 34\}, \{43, 28\}, \{44, 19\}, \{45, 39\}, \{46, 33\}, \{47, 27\}, \{48, 14\}, \{49, 38\}, \{50, 32\}, \{51, 20\}, \{52, 15\}, \{53, 37\}, \{54, 26\}, \{55, 21\}, \{56, 16\}, \{57, 31\}, \{58, 25\}, \{59, 22\}\}$. ■

VII. ASYMPTOTIC ANALYSIS OF ARRAY CODES

The goal of this section is to provide a figure of merit in order to compare between the different constructions of array codes. For simplicity we consider the case where $\ell = t$, that is, it is possible to read all the bits in every bucket. Under this setup, it holds that $FP_{t,t}(s, k) \leqslant sk/t$ for all s, k, and t. This motivates us to define the following values

$$\mathcal{R}_X(t, k) = \limsup_{s\to\infty} \frac{X_{t,t}(s, k)}{sk/t}, \tag{4}$$

where $X \in \{P, B, FP, FB\}$. Since array codes have different parameters, the goal of this figure of merit is to have a measure that will allow for a fair comparison between different code constructions. This is accomplished by removing the dependency on the number of bits s while fixing the values of k and t as done in Equation (4). In many of the existing constructions the value of s is fixed and constructions for a larger number of bits are achieved by concatenation of the smaller blocks of a smaller number of bits. Thus, for fixed values of k and t it is possible to compare between two code constructions even if they have a different number of information bits.

The case where $t = 1$ has been studied in several previous works. For example, for functional PIR array codes we have $\mathcal{R}_{FP}(1, k) \geqslant \frac{1}{k \cdot H(1/k)}$ for any even integer $k \geqslant 4$ [51, Th. 14]. Also, for functional batch array codes it holds from [51, Th. 22] that $\mathcal{R}_{FB}(1, k) \leqslant \frac{1}{k \cdot H(c_k)}$, where $c_1 = \frac{1}{2}$ and c_{k+1} is the root of the polynomial $H(z) = H(c_k) - zH(c_k)$. The function $H(\cdot)$ denotes the binary entropy function defined by $H(p) = -p \log p - (1 - p) \log (1 - p)$. For the case $k = 1$ we have $\mathcal{R}_{FB}(t, 1) = \mathcal{R}_{FP}(t, 1) = 1$ from Theorem 16(b). According to the bounds and constructions studied in the paper, we can already summarize several results in the following theorems for $t = 2$ and general values.

Theorem 35:

a) $\mathcal{R}_{FP}(2, 2) \leqslant \mathcal{R}_{FB}(2, 2) \leqslant \frac{7}{8} = 0.875$, and $\mathcal{R}_{FB}(2, 2) \geqslant 0.71$.
b) $\mathcal{R}_{FP}(2, 11) \leqslant \frac{25}{33} = 0.758$.
c) $\mathcal{R}_{FP}(2, 14) \leqslant \frac{25}{28} = 0.893$.
d) $\mathcal{R}_{FP}(2, 48) \leqslant \frac{3}{4} = 0.75$.
e) $\mathcal{R}_P(2, 16) \leqslant \frac{25}{32} = 0.78125$.
f) $\mathcal{R}_B(2, 15) \leqslant \frac{5}{9} = 0.556$.

Proof:

a) From Theorem 18 we have $FB_{2,2}(s, 2) \leqslant 7 \cdot \lceil \frac{s}{8} \rceil$. Thus, $\mathcal{R}_{FB}(2, 2) = \limsup_{s\to\infty} \frac{FB_{2,2}(s,2)}{2s/2} \leqslant \limsup_{s\to\infty} \frac{7\lceil s/8 \rceil}{s} \leqslant \limsup_{s\to\infty} \frac{(7s/8)+7}{s} = \frac{7}{8}$.
From Corollary 19 we have $FB_{2,2}(s, 2) \geqslant 0.71s$. Thus, $\mathcal{R}_{FB}(2, 2) = \limsup_{s\to\infty} \frac{FB_{2,2}(s,2)}{2s/2} \geqslant \limsup_{s\to\infty} \frac{0.71s}{s} = 0.71$.
b) From Theorem 26 we have $FP_{2,2}(6, 11) \leqslant 25$. Then, it is possible to use Theorem 3(e) to get that $FP_{2,2}(s, 11) \leqslant 25 \cdot \lceil \frac{s}{6} \rceil$. Thus, $\mathcal{R}_{FP}(2, 11) = \limsup_{s\to\infty} \frac{FP_{2,2}(s,11)}{11s/2} \leqslant \limsup_{s\to\infty} \frac{25\lceil s/6 \rceil}{11s/2} \leqslant \limsup_{s\to\infty} \frac{(25s/6)+25}{11s/2} = \frac{50}{66} = 0.758$.
c) From Theorem 33 we have $FP_{2,2}(4, 14) \leqslant 25$. Then, it is possible to use Theorem 3(e) to get that $FP_{2,2}(s, 14) \leqslant 25 \cdot \lceil \frac{s}{4} \rceil$. Thus, $\mathcal{R}_{FP}(2, 14) = \limsup_{s\to\infty} \frac{FP_{2,2}(s,14)}{14s/2} \leqslant \limsup_{s\to\infty} \frac{25\lceil s/4 \rceil}{7s} \leqslant \limsup_{s\to\infty} \frac{(25s/4)+25}{7s} = \frac{25}{28} = 0.893$.
d) From Theorem 34 we have $FP_{2,2}(5, 48) \leqslant 90$. Then, it is possible to use Theorem 3(e) to get that

$FP_{2,2}(s,48) \leqslant 90 \cdot \lceil \frac{s}{5} \rceil$. Thus, $\mathcal{R}_{FP}(2,48) = \limsup_{s\to\infty} \frac{FP_{2,2}(s,48)}{48s/2} \leqslant \limsup_{s\to\infty} \frac{90\lceil s/5 \rceil}{24s} \leqslant \limsup_{s\to\infty} \frac{(90s/5)+90}{24s} = \frac{90}{120} = \frac{3}{4} = 0.75$.

e) From Theorem 32 we have $P_{2,1}(4,16) \leqslant 25$. Then, it is possible to use Theorem 3(e) and get that $P_{2,1}(s,16) \leqslant 25 \cdot \lceil \frac{s}{4} \rceil$. Thus, $\mathcal{R}_P(2,16) = \limsup_{s\to\infty} \frac{P_{2,2}(s,16)}{16s/2} \leqslant \limsup_{s\to\infty} \frac{25\lceil s/4 \rceil}{8s} \leqslant \limsup_{s\to\infty} \frac{(25s/4)+25}{8s} = \frac{25}{32} = 0.78125$.

f) From Theorem 25 we have $B_{2,2}(6,15) = 25$. Then, it is possible to use Theorem 3(e) and get that $B_{2,2}(s,15) \leqslant 25 \cdot \lceil \frac{s}{6} \rceil$. Thus, $\mathcal{R}_B(2,15) = \limsup_{s\to\infty} \frac{B_{2,2}(s,15)}{15s/2} \leqslant \limsup_{s\to\infty} \frac{25\lceil s/6 \rceil}{15s/2} \leqslant \limsup_{s\to\infty} \frac{(25s/6)+25}{15s/2} = \frac{25}{45} = 0.556$.

■

Theorem 36:

a) For any $r \geqslant 3$, $\mathcal{R}_P(r^2-r+1,r) \leqslant \frac{(r+1)(r^2-r+1)}{r(r^2+r)}$ (also for B).

b) For any $t \geqslant 2$, $\mathcal{R}_P(t,k) \leqslant \frac{m}{k(t+1)}$, where $k = \binom{t(t+1)}{t}$ and $m = k + \frac{\binom{t(t+1)}{t+1}}{t}$.

c) For any two integers t and k, $\mathcal{R}_{FB}(t,k) \leqslant \frac{1}{k \cdot H(c_{tk})}$, where $c_1 = \frac{1}{2}$ and c_{k+1} is the root of the polynomial $H(z) = H(c_k) - zH(c_k)$.

d) For any positive integers t,k and a, $\mathcal{R}_X(t,a\cdot k) \leqslant \mathcal{R}_X(t,k)$, where $X \in \{P,B,FP,FP\}$.

e) For any positive integers t,k and a, $\mathcal{R}_X(t,k) \leqslant \mathcal{R}_X(a\cdot t,k)$, where $X \in \{P,B,FP,FP\}$.

Proof:

a) From Theorem 28 we have for any $r \geqslant 3$, $P_{r^2-r+1,r-1}(r^2+r,r) \leqslant r+1$. Then, it is possible to use Theorem 3(e) to get that $P_{r^2-r+1,r-1}(s,r) \leqslant (r+1) \cdot \left\lceil \frac{s}{r^2+r} \right\rceil$. Thus, for a given r, it holds that

$$\begin{aligned}\mathcal{R}_P(r^2-r+1,r) &= \limsup_{s\to\infty} \frac{P_{r^2-r+1,r^2-r+1}(s,r)}{rs/(r^2-r+1)} \\ &\leqslant \limsup_{s\to\infty} \frac{P_{r^2-r+1,r-1}(s,r)}{rs/(r^2-r+1)} \\ &\leqslant \limsup_{s\to\infty} \frac{(r+1)\cdot\left\lceil \frac{s}{r^2+r} \right\rceil}{rs/(r^2-r+1)} \\ &\leqslant \limsup_{s\to\infty} \frac{\frac{(r+1)s}{r^2+r}+(r+1)}{rs/(r^2-r+1)} = \frac{(r+1)(r^2-r+1)}{r(r^2+r)}.\end{aligned}$$

b) From Theorem 2(e) we have for any $t \geqslant 2$ and $p = t+1$, $P_{t,t}(t(t+1),k) \leqslant m$, where $k = \binom{t(t+1)}{t}$ and $m = k + \frac{\binom{t(t+1)}{t+1}}{t}$. Then, it is possible to use Theorem 3(e) to get that $P_{t,t}(s,k) \leqslant m \cdot \left\lceil \frac{s}{t(t+1)} \right\rceil$. Thus, for a given t, it holds that $\mathcal{R}_P(t,k) = \limsup_{s\to\infty} \frac{P_{t,t}(s,k)}{sk/t} \leqslant \limsup_{s\to\infty} \frac{m\cdot\lceil \frac{s}{t(t+1)} \rceil}{sk/t} \leqslant \limsup_{s\to\infty} \frac{\frac{m\cdot s}{t(t+1)}+m}{sk/t} = \frac{m}{k(t+1)}$.

c) From Lemma 10, we have $FB_{t,t}(s,k) \leqslant FB_{t,1}(s,k) \leqslant FB(\lceil s/t \rceil, t\cdot k)$. $\mathcal{R}_{FB}(t,k) = \limsup_{s\to\infty} \frac{FB_{t,t}(s,k)}{sk/t} \leqslant \limsup_{s\to\infty} \frac{FB(\lceil s/t \rceil,t\cdot k)}{sk/t} = \limsup_{s\to\infty} \frac{FB(\lceil s/t \rceil,t\cdot k)}{s/t} \cdot \frac{1}{k}$. Thus, according to [51, Th. 22], $\mathcal{R}_{FB}(t,k) \leqslant \limsup_{s\to\infty} \frac{FB(\lceil s/t \rceil,t\cdot k)}{s/t} \cdot \frac{1}{k} \leqslant \frac{1}{k\cdot H(c_{tk})}$, where $c_1 = \frac{1}{2}$ and c_{k+1} is the root of the polynomial $H(z) = H(c_k) - zH(c_k)$.

d) From Theorem 3(c) we have that for any positive integer a and any $X \in \{P,B,FP,FP\}$, $X_{t,t}(s,a\cdot k) \leqslant a \cdot X_{t,t}(s,k)$. Thus, $R_X(t,a\cdot k) = \limsup_{s\to\infty} \frac{X_{t,t}(s,a\cdot k)}{ska/t} \leqslant \limsup_{s\to\infty} \frac{a\cdot X_{t,t}(s,k)}{ska/t} = \limsup_{s\to\infty} \frac{X_{t,t}(s,k)}{sk/t} = \mathcal{R}_X(t,k)$.

e) From Theorem 3(f) we have that for any positive integer a and any $X \in \{P,B,FP,FP\}$, $a \cdot X_{a\cdot t,a\cdot t}(s,k) \geqslant X_{t,a\cdot t}(s,k) = X_{t,t}(s,k)$. Thus, $R_X(t,k) = \limsup_{s\to\infty} \frac{X_{t,t}(s,k)}{sk/t} \leqslant \limsup_{s\to\infty} \frac{a\cdot X_{a\cdot t,a\cdot t}(s,k)}{sk/t} = \limsup_{s\to\infty} \frac{X_{a\cdot t,a\cdot t}(s,k)}{sk/(at)} = \mathcal{R}_X(a\cdot t,k)$.

■

We analyzed the results in this paper mostly for the case of $t=2$ in Theorem 35 to find the best constructions for several values of k. In Theorem 36 several more results were derived. We see this figure of merit as a comparison measure for more future constructions.

We can think about $\mathcal{R}_X(t,k)$ as the rate of the code. Due to the fact that our aim is to find the minimum value of $X_{t,t}(s,k)$, then, a smaller value of $\mathcal{R}_X(t,k)$ indicates a better construction. For example, from Theorem 35 we can conclude that the construction that leads to the result of Theorem 35(d) is better than the constructions that lead to the results of Theorem 35(a)(b)(c).

VIII. Locality Codes

In this section we study a new family of array codes which is a special case of functional PIR array codes in the sense that each recovering set is of size at most r and all the cells of each bucket can be read, i.e., $\ell = t$. This new family of array codes will be called *locality functional array codes*. In order to find lower bounds and constructions for locality functional array codes we will use codes and designs in subspaces and covering codes.

A. Definitions and Basic Constructions

This section is studying the following family of codes.

Definition 37: An (s,k,m,t,r) **locality functional array code** over Σ is defined by an encoding map $\mathcal{E}: \Sigma^s \to (\Sigma^t)^m$ that encodes s information bits $x_1,\ldots,x_s$ into a $t\times m$ array and a decoding function $\mathcal{D}$ that satisfies the following property. For any request of a linear combination $\boldsymbol{v}$ of the information bits, there is a partition of the columns into k recovering sets $S_1,\ldots,S_k \subseteq [m]$ where $|S_j| \leqslant r$ for any $j \in [k]$.

We denote by $D(s,k,t,r)$ the smallest number of buckets m such that an (s,k,m,t,r) locality functional array code exists. For the rest of the section, assume that the parameters s,k,t and r are positive integers such that $t \leqslant s$. The following theorem summarizes several results on $D(s,k,t,r)$ based upon basic bound and constructions.

Theorem 38:

a) $D(s,k,t,r) \geqslant m^*$, where m^* is the smallest positive integer such that $\sum_{i=1}^{\min\{r,m^*-k+1\}} \binom{m^*}{i}(2^t-1)^i \geqslant k(2^s-1)$.

b) For any integer a where $1 \leqslant a < t$, $D(s,k,t,r) \leqslant D(s-a,k,t-a,r)$.

c) For every positive integers s_1, s_2, r_1, r_2 and p, $D(s_1+s_2,k,t,r_1+r_2) \leqslant D(s_1,k,t,r_1)+D(s_2,k,t,r_2)$. In particular, $D(ps,k,t,pr) \leqslant p \cdot D(s,k,t,r)$.

Proof:

a) Similar to the proof of Theorem 7 but with minor changes. Here, all cells from each bucket can be read. Hence, for any positive integer n, there are $(2^t-1)^n$ nonzero linear combinations that can be obtained from n buckets while using all the n buckets. Also, each recovering set must be of size at most $\min\{r, m^*-k+1\}$. Thus, we get that $\sum_{i=1}^{\min\{r,m^*-k+1\}} \binom{m^*}{i}(2^t-1)^i \geqslant k(2^s-1)$.

b) Let $\mathcal{C}$ be an $(s-1,k,m,t-1,r)$ locality functional array code with m buckets such that each bucket has $t-1$ cells. For the s information bits $x_1,\ldots,x_s$, we encode the first $s-1$ bits using the encoder of $\mathcal{C}$ to get m buckets where each bucket has $t-1$ cells. For each bucket, a new cell that stores x_s is added. Assume that R is the request which is a linear combination of the s information bits. Let R_1 be the part of the request which is a linear combination of the first $s-1$ information bits. From the properties of $\mathcal{C}$, for the request R_1, there exist k disjoint recovering sets $\{\mathcal{S}_1,\mathcal{S}_2,\ldots,\mathcal{S}_k\}$ such that $|\mathcal{S}_j| \leqslant r$ for any $j \in [k]$. If $R = R_1$, then the same $\{\mathcal{S}_1,\mathcal{S}_2,\ldots,\mathcal{S}_k\}$ are recovering sets for R. If $R = x_s$, we can take the first k buckets as k recovering sets each of size 1. If R includes x_s, then the same $\{\mathcal{S}_1,\mathcal{S}_2,\ldots,\mathcal{S}_k\}$ are recovering sets for R, where we can read x_s from one of the buckets in each $\mathcal{S}_j$. Thus, $D(s,k,t,r) \leqslant D(s-1,k,t-1,r)$ and we can get that $D(s,k,t,r) \leqslant D(s-a,k,t-a,r)$ by induction on a.

c) Let $\mathcal{C}_1$ be an (s_1,k,m_1,t,r_1) locality functional array code and $\mathcal{C}_2$ be an (s_2,k,m_2,t,r_2) locality functional array code. The codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are used to construct an $(s_1+s_2,k,m_1+m_2,t,r_1+r_2)$ locality functional array code by encoding the first s_1 bits using the encoder of $\mathcal{C}_1$ and the last s_2 bits using the encoder of $\mathcal{C}_2$. Assume that R is the request which is a linear combination of the s_1+s_2 information bits. Let R_1, R_2 be the part of R which is a linear combination of the first s_1, last s_2 information bits, respectively. According to $\mathcal{C}_1, \mathcal{C}_2$, there exist k recovering sets $\{\mathcal{S}_1^1,\mathcal{S}_2^1,\ldots,\mathcal{S}_k^1\}, \{\mathcal{S}_1^2,\mathcal{S}_2^2,\ldots,\mathcal{S}_k^2\}$ for R_1, R_2 such that each recovering set has size at most r_1, r_2, respectively. Then, the set $\mathcal{S}_j^1 \cup \mathcal{S}_j^2$ for any $j \in [k]$ is a recovering set for R with size at most r_1+r_2. Therefore, the sets $\{\mathcal{S}_1^1 \cup \mathcal{S}_1^2, \mathcal{S}_2^1 \cup \mathcal{S}_2^2, \ldots, \mathcal{S}_k^1 \cup \mathcal{S}_k^2\}$ are k recovering sets for R such that the size of each recovering set is at most r_1+r_2. Thus, $D(s_1+s_2,k,t,r_1+r_2) \leqslant D(s_1,k,t,r_1)+D(s_2,k,t,r_2)$ and we can get that $D(ps,k,t,pr) \leqslant p \cdot D(s,k,t,r)$ by induction on p.

■

B. Constructions Based on Subspaces

In this section we show connections between the problem of finding the minimal number of buckets for locality functional array codes and several problems in subspaces. Subspaces were used in [34] to construct *array codes* and to examine their locality and availability. The family of array codes that was defined in [34] is a linear subspace of $b \times n$ matrices over $\mathbb{F}_q$ such that each codeword is a $b \times n$ matrix where each entry is called a symbol. The *weight* of each codeword was defined to be the number of nonzero columns in the codeword and the distance of the code is the minimal weight of a nonzero codeword.

The problem that was presented in [34] was to examine locality and availability of array codes where two types of locality were defined. The first one is *node locality*. A codeword column $j \in [n]$ has node locality r_{nd} if it can be recovered by a linear combination of the symbols of the columns in a recovering set of size r_{nd}. If all codeword columns have node locality r_{nd}, then r_{nd} is also called the node locality of the array code. The second type is *symbol locality* r_{sb} which is similar to node locality but instead of recovering the whole column, here only one symbol (entry of the codewords matrices) is needed to be recovered. Similarly, there are two types of availability. The node, symbol availability, denoted by t_{nd}, t_{sb} is the number of pairwise disjoint recovering sets of size at most r_{nd}, r_{sb} for any codeword column, symbol, respectively.

To simplify the problem, they flattened each $b \times n$ codeword into a vector of length bn by reading the symbols of the codeword column by column from first to last entry. The $M \times bn$ generator matrix G, where each row is a flattened codeword, can represent the array code C, where the columns $(j-1)b+1,\ldots,jb$ of G correspond to the symbols of the j-th codeword column of C and these columns are called the j-th *thick column* of G. By this way, the j-th thick column of G which corresponds to the j-th codeword column of C, can be represented by V_j which is a b-subspace of $\mathbb{F}_q^M$. Thus, an equivalent constraints of node and symbol locality can be formed using subspaces as stated in [34, Lemma 3], where a subset $\mathcal{S} = \{j_1,\ldots,j_p\} \subseteq [n] \setminus \{j\}$ is a recovering set for the codeword column $j \in [n]$, if and only if $V_j \subseteq V_{j_1}+\cdots+V_{j_p}$. Similarly, $\mathcal{S}$ is a recovering set for the symbol $(i,j), i \in [b], j \in [n]$ if and only if $\boldsymbol{g}_{(j-1)b+i} \in V_{j_1}+\cdots+V_{j_p}$, where $\boldsymbol{g}_{(j-1)b+i}$ is the i-th column in the j-th thick column of G that corresponds to the i-th entry in the j-th codeword column of C.

In our work we are interested in the problem of recovering the requests which are all possible linear combinations of the information bits, which is different from the problem in [34] where the nodes or symbols that are part of the code are needed to be recovered. We can apply some of the results and constructions from [34] in our case. Recall that we defined $\Sigma = \mathbb{F}_2$. Let Σ^s be a vector space of dimension s over Σ. We can consider each bucket which has t cells, as a subspace of Σ^s with dimension t and denote a subspace of dimension t as a *t-subspace*. The following claim is motivated by [34, Lemma 3].

Claim 1: The value of $D(s,k,t,r)$ is the smallest number m of t-subspaces of Σ^s such that there exists a partition of the subspaces into k subsets, $\mathcal{S}_1,\ldots,\mathcal{S}_k$, that satisfies the following property. The size of each subset $\mathcal{S}_i$ is at most

r and for every request R, which can be represented by a 1-subspace W, it holds that for each $\mathcal{S}_i$, $W \subseteq \Sigma_{j=0}^{r'} \mathcal{S}_{i_j}$ where $\mathcal{S}_{i_j}$ is the j-th subspace in $\mathcal{S}_i$ and $|\mathcal{S}_i| = r' \leqslant r$.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$ be the vector of dimension $1 \times s$ with the s information bits and let V be a t-subspace of Σ^s. It is said that a bucket with t cells *stores* a t-subspace V if for a given basis $\mathcal{B} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_t\}$, the i-th cell $i \in [t]$ of the bucket stores the linear combination $\langle \boldsymbol{v}_i, \boldsymbol{x} \rangle$. Note that the choice of the basis $\mathcal{B}$ does not matter and we can choose any basis of V. Each request R which is a linear combination of the s information bits can be represented by a 1-subspace W of Σ^s. It is said that a request is *contained* in a bucket b if the set $\{b\}$ is a recovering set for the request. Note that if W is contained in a t-subspace V then the request R is contained in the bucket that stores V.

Let $\mathcal{G}_q(s,t)$ denote the set of all t-dimensional subspaces of the vector space $\mathbb{F}_q^s$. The set $\mathcal{G}_q(s,t)$ is often called the *Grassmannian* [18]. It is well known that

$$|\mathcal{G}_q(s,t)| = \begin{bmatrix} s \\ t \end{bmatrix}_q := \frac{(q^s-1)(q^{s-1}-1)\cdots(q^{s-t+1}-1)}{(q^t-1)(q^{t-1}-1)\cdots(q-1)},$$

where $\begin{bmatrix} s \\ t \end{bmatrix}_q$ is the q-ary Gaussian coefficient [40]. The following is a definition of *spreads* from [21] which are partitions of vector spaces.

Definition 39: Let $s = at$. Then a set $\mathcal{S} \subseteq \mathcal{G}_q(s,t)$ is called a t-**spread** if all elements of $\mathcal{S}$ intersect only trivially and they cover the whole space $\mathbb{F}_q^s$.

It is known that the size of a t-spread of $\mathbb{F}_q^s$ is $\frac{q^s-1}{q^t-1}$ when s is a multiple of t [21]. It is also follows that spreads do not exist when t does not divide s. In case s is not a multiple of t there is a notion of *partial spreads*, where a *partial t-spread* of $\mathbb{F}_q^s$ is a collection of mutually disjoint t-subspaces. For the problem we are studying in this section, partial spreads cannot be used due to the fact that they do not necessarily cover the whole space. Thus, in order to deal with the cases when t does not divide s we use *covering designs* which are defined as follows [17].

Definition 40: A **covering design** $\mathbb{C}_q(s,t,a)$ is a subset $\mathcal{S} \subseteq \mathcal{G}_q(s,t)$ such that each element of $\mathcal{G}_q(s,a)$ is contained in at least one subspace from $\mathcal{S}$.

The covering number $C_q(s,t,a)$ is the minimum size of a covering design $\mathbb{C}_q(s,t,a)$. From [17, Th. 4.6] we get that for any $1 \leqslant t \leqslant s$,

$$C_q(s,t,1) = \left\lceil \frac{q^s-1}{q^t-1} \right\rceil. \tag{5}$$

Note that when $t|s$, an optimal covering design $\mathbb{C}_q(s,t,1)$ is exactly a t-spread of $\mathbb{F}_q^s$. Now, we will define another family of partitions and another family of codes that can be used to construct locality functional array codes. The following is a definition of λ-*fold partitions* from [16].

Definition 41: Let λ be a positive integer. A λ-**fold partition** of the vector space $V = \mathbb{F}_q^s$ is a multiset $\mathcal{S}$ of subspaces of V such that every nonzero vector in V is contained in exactly λ subspaces in $\mathcal{S}$.

Note that a 1-fold partition of $\mathbb{F}_q^s$ that does not contain a subspace with dimension larger than t is also a covering design $\mathbb{C}_q(s,t,1)$. Denote by $A_q(s,t,\lambda)$ the minimum size of a λ-fold partition of $\mathbb{F}_q^s$ that does not contain a subspace with dimension larger than t. In [16], it is also possible to find results on λ-fold partitions. For example, there exists a construction of a $\left(\frac{2^t-1}{2^p-1}\right)$-fold partition of Σ^s with $\frac{2^s-1}{2^p-1}$ t-subspaces where $p = \gcd(s,t)$. Therefore, $A_2(s,t,\frac{2^t-1}{2^p-1}) \leqslant \frac{2^s-1}{2^p-1}$.

Lastly, the following is a definition of *covering Grassmannian codes* from [18].

Definition 42: For every positive integers α and δ where $\delta + t \leqslant s$, an α-$(s,t,\delta)_q^c$ **covering Grassmannian code** $\mathbb{C}$ is a subset of $\mathcal{G}_q(s,t)$ such that each subset of α codewords of $\mathbb{C}$ spans a subspace whose dimension is at least $\delta + t$ in $\mathbb{F}_q^s$.

The value $B_q(s,t,\delta;\alpha)$ will denote the maximum size of an α-$(s,t,\delta)_q^c$ covering Grassmannian code. The following theorem summarizes some bounds on $D(s,k,t,r)$ using spreads, covering designs, λ-fold partitions, and covering Grassmannian codes.

Theorem 43: For each s,t,k and r positive integers

a) $D(s,1,t,1) = C_2(s,t,1) = \left\lceil \frac{2^s-1}{2^t-1} \right\rceil$.
b) $D(s,1,t,r) \leqslant r \cdot \left\lceil \frac{2^{s/r}-1}{2^t-1} \right\rceil$, where $r|s$.
c) $D(s,k,t,1) \leqslant A_2(s,t,k)$.
d) $D(s, \lfloor B_2(s,t,s-t;r)/r \rfloor, t, r) \leqslant B_2(s,t,s-t;r)$.
e) $D(s, \begin{bmatrix} s-1 \\ t-1 \end{bmatrix}_2, t, 1) \leqslant \begin{bmatrix} s \\ t \end{bmatrix}_2$, where $t > 1$.
f) $D(s, \left\lfloor \frac{2^s-2^t}{r \cdot 2^t - r} \right\rfloor + 1, t, r) \leqslant \frac{2^s-1}{2^t-1}$, where $s = rt$.

Proof:

a) To prove this part we use a construction motivated by [34, Construction 2]. Let $\mathbb{C}$ be a $\mathbb{C}_2(s,t,1)$ covering design with $C_2(s,t,1)$ t-subspaces. To construct an $(s, C_2(s,t,1), t, 1)$ locality functional array code, we take $C_2(s,t,1)$ buckets where each bucket stores one of the t-subspace from $\mathbb{C}$. From Definition 40, every 1-subspace of Σ^s is contained in at least one t-subspace from $\mathbb{C}$. Thus, each request R which can be represented by a 1-subspace of Σ^s, is contained in at least one bucket. Therefore, by using Equation (5) we get that $D(s,1,t,1) \leqslant C_2(s,t,1) = \left\lceil \frac{2^s-1}{2^t-1} \right\rceil$.
For the other direction, assume that $\mathcal{C}$ is an $(s,1,m,t,1)$ locality functional array code with m buckets. We construct a $\mathbb{C}_2(s,t,1)$ covering design with m t-subspaces of Σ^s that are stored in the m buckets of $\mathcal{C}$. Let W be a 1-subspace of Σ^s that represents a request R for the code $\mathcal{C}$. From the property of the code $\mathcal{C}$, there exists one bucket that contains R. Therefore, there exists one t-subspace in $\mathbb{C}$ that contains W. Thus, $C_2(s,t,1) \leqslant D(s,1,t,1)$.
b) This result holds from part (a) in this theorem and Theorem 38(c).
c) Let $\mathcal{S}$ be a k-fold partition of Σ^s that does not contain a subspace with dimension larger than t. Assume that $|\mathcal{S}| = m$. To construct a locality functional array code, we take m buckets where each bucket stores one of the subspaces from $\mathcal{S}$. Assume that R is the request which can be represented by a vector $\boldsymbol{u}$ of Σ^s. Then, from the property of the multiset $\mathcal{S}$, the vector $\boldsymbol{u}$ is

contained in exactly k subspaces in $\mathcal{S}$. Therefore, R is contained in exactly k buckets. Thus, the m buckets form an $(s,k,m,t,1)$ locality functional array code, and hence, $D(s,k,t,1) \leqslant A_2(s,t,k)$.

d) Let $\mathbb{C}$ be an r-$(s,t,s-t)_2$ covering Grassmannian code with m t-subspaces of Σ^s. We take m buckets where each bucket stores one of the t-subspaces from $\mathbb{C}$. Let R be the request. From the property of the code $\mathbb{C}$, every subset of r t-subspaces of $\mathbb{C}$ spans the whole space Σ^s. Hence, every subset of r buckets contains R. Therefore, we can partition the m buckets into $\lfloor m/r \rfloor$ parts, where each part contains R, and hence, there exist $\lfloor m/r \rfloor$ recovering sets for R. Thus, the construction with the m buckets forms an $(s, \lfloor m/r \rfloor, m, t, r)$ locality functional array code.

e) To prove this part we use a construction motivated by [34, Construction 1]. We construct an $(s, \begin{bmatrix} s-1 \\ t-1 \end{bmatrix}_2, \begin{bmatrix} s \\ t \end{bmatrix}_2, t, 1)$ locality functional array code by taking $\begin{bmatrix} s \\ t \end{bmatrix}_2$ buckets where each bucket has t cells and stores one of the t-subspaces of Σ^s. Every 1-subspace of Σ^s is contained in exactly $\begin{bmatrix} s-1 \\ t-1 \end{bmatrix}_2$ t-subspaces. Therefore, every request R which can be represented by a 1-subspace is contained in exactly $\begin{bmatrix} s-1 \\ t-1 \end{bmatrix}_2$ buckets. Thus, we get that $D(s, \begin{bmatrix} s-1 \\ t-1 \end{bmatrix}_2, t, 1) \leqslant \begin{bmatrix} s \\ t \end{bmatrix}_2$.

f) Let $s = rt$ and $\mathcal{S}$ be a t-spread of Σ^s such that $|\mathcal{S}| = \frac{2^s-1}{2^t-1}$. To construct a locality functional array code we store each t-subspace in $\mathcal{S}$ in a bucket with t cells. Assume that R is the request which can be represented by a 1-subspace W of Σ^s. From the property of spreads, there exists a subspace in $\mathcal{S}$ that includes W. Therefore, there exists a bucket that contains R which can form a recovering set of size 1. Then, partition the remaining $\frac{2^s-1}{2^t-1} - 1 = \frac{2^s-2^t}{2^t-1}$ buckets into $\left\lfloor \frac{2^s-2^t}{r \cdot 2^t - r} \right\rfloor$ parts where each part has size r. Each part $\mathcal{P}_i$ has r mutually disjoint t-subspaces $U_{i_1}, U_{i_2}, \ldots, U_{i_r}$. Hence, $\sum_{j=1}^{r} U_{i_j} = \Sigma^s$. Thus, each part $\mathcal{P}_i$ is a recovering set of R of size r. Then, there exist $1 + \left\lfloor \frac{2^s-2^t}{r \cdot 2^t - r} \right\rfloor$ recovering sets each of size at most r and the code is an $(s, \left\lfloor \frac{2^s-2^t}{r \cdot 2^t - r} \right\rfloor + 1, \frac{2^s-1}{2^t-1}, t, r)$ locality functional array code.

■

The following is an example of Theorem 43(c).

Example 4: In this example we will use an example of a 2-fold partition from [16] in order to construct a locality functional array code. Let $s = 3$. The following multiset $\mathcal{S}$ of subspaces of Σ^3 is a 2-fold partition that does not contain a subspace with dimension larger than $t = 2$.

$\mathcal{S} = \{\{100, 011, 111\}, \{010, 001, 011\}, \{001, 110, 111\}, \{110, 010, 100\}, \{101\}, \{101\}\}$.

We represent each element in Σ^3 as a binary vector of length 3 and every subspace in $\mathcal{S}$ by its elements except the zero vector. It holds that any binary vector of length 3 is contained in exactly two subspaces in $\mathcal{S}$, and hence, $A_2(3,2,2) \leqslant 6$. We construct a $(3,2,6,2,1)$ locality functional array code with the following buckets that are obtained from $\mathcal{S}$.

1	2	3	4	5	6
x_3	x_2	x_1	x_2x_3	x_1x_3	x_1x_3
x_1x_2	x_1	x_2x_3	x_2		

For example, if the request is $x_1 + x_2$, then the recovering sets are $\{\{1\}, \{2\}\}$.

The following is an example of Theorem 43(f).

Example 5: For $s = 4, t = 2$ and $r = 2$, the following set $\mathcal{S}$ is a 2-spread of Σ^4 of size $\frac{2^4-1}{2^2-1} = 5$.

$\mathcal{S} = \{\{0001, 0010\}, \{0100, 1000\}, \{0101, 1010\}, \{1001, 0111\}, \{0110, 1011\}\}$.

We represent each element in Σ^4 as a binary vector of length 4 and every 2-subspace as a basis with 2 vectors. We construct a $(4,3,5,2,2)$ locality functional array code with the following buckets that are obtained from $\mathcal{S}$.

1	2	3	4	5
x_1	x_3	x_1x_3	x_1x_4	x_2x_3
x_2	x_4	x_2x_4	$x_1x_2x_3$	$x_1x_2x_4$

For example, if the request is $x_1 + x_2$, then the recovering sets are $\{\{1\}, \{2,3\}, \{4,5\}\}$.

C. Bounds and Constructions Based Upon Covering Codes

In this section we show how covering codes are used to construct locality functional array codes and to get lower bounds for $D(s,k,t,r)$. For the rest of the section we assume that $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$ is the vector of dimension $1 \times s$ with the s information bits. For the case of $t = 1$ the following result can be obtained. Remember that $h[s,r]_q$ is the smallest length of a linear covering code over $\mathbb{F}_q$ with covering radius r and redundancy s.

Theorem 44: $D(s,1,1,r) = h[s,r]$.

Proof: There exists an $[h[s,r], h[s,r] - s, r]$ linear covering code with some parity check matrix H. To construct a locality functional array code we store in each bucket the linear combination $\langle \boldsymbol{h}_i, \boldsymbol{x} \rangle$ where $\boldsymbol{h}_i$ is the i-th column of H. Assume that R is the request which can be represented by a binary vector $\boldsymbol{u} \in \Sigma^s$. From Property 13, we know that the vector $\boldsymbol{u}$ can be represented as the sum of at most r columns of H. Therefore, there exists a recovering set of size at most r for the request R. The number of buckets is the number of columns of H which is $h[s,r]$. Thus, $D(s,1,1,r) \leqslant h[s,r]$. The lower bound can be obtained from Corollary 56 which will appear later. ■

We can generalize the connection of covering codes and locality functional array codes with general t. We start by defining a partition of matrices.

Definition 45: A **t-partition** of a matrix H is a collection $\mathcal{P}$ of subspaces of dimension t with the property that every column vector of H is contained in at least one member of $\mathcal{P}$.

A t-partition is called **strict** if every column vector of H is contained in exactly one member of $\mathcal{P}$.

The next theorem shows the connection between covering codes and locality functional array codes with $k = 1$.

Theorem 46: Let H be a parity check matrix for an $[n, n-s, r]$ covering code, and let p be the smallest size of a t-partition of H. Then, $D(s, 1, t, r) \leqslant p$.

Proof: Let H be a parity check matrix of a given $[n, n-s, r]$ covering code. Let $\mathcal{P}$ be a t-partition of H, that contains p subspaces of dimension t. We construct an $(s, 1, p, t, r)$ locality functional array code $\mathcal{C}$ by storing each t-subspace from $\mathcal{P}$ in one bucket with t cells. Let $\boldsymbol{u} \in \Sigma^s$ be a request which represents the linear combination $\langle \boldsymbol{u}, \boldsymbol{x} \rangle$ of the s information bits. From Property 13, we know that there exists a vector $\boldsymbol{y} \in \Sigma^n$ such that $H \cdot \boldsymbol{y} = \boldsymbol{u}$, where $w = w_H(\boldsymbol{y}) \leqslant r$. If $w_H(\boldsymbol{y}) = r' \leqslant r$, then the request $\boldsymbol{u}$ is equal to the sum of r' columns of H and denote them by $\boldsymbol{h}_{i_1}, \boldsymbol{h}_{i_2}, \ldots, \boldsymbol{h}_{i_{r'}}$. We know that each 1-subspace with a basis $\{\boldsymbol{h}_{i_j}\}, j \in [r']$ is contained in one subspace from the partition $\mathcal{P}$, and hence, the vector $\boldsymbol{h}_i$ is contained in one bucket of $\mathcal{C}$. Thus, we can get all the r' columns from at most $r' \leqslant r$ buckets. ∎

Now, the method to get locality functional array codes from covering codes over $\mathbb{F}_q$ is established. We follow an example from [7] and for that we use the following definition in the rest of this section.

Definition 47: Let $\mathcal{B} = \{1, \epsilon, \epsilon^2, \ldots, \epsilon^{w-1}\}$ be a basis for $\mathbb{F}_{2^w}$ over Σ where ϵ is a primitive element of $\mathbb{F}_{2^w}$. For each $i \in [0, 2^w - 2]$ let $(\epsilon^i)_w$ be the binary column vector of length w that represents the element ϵ^i of $\mathbb{F}_{2^w}$ with respect to the basis $\mathcal{B}$. Let $\mathcal{U}_0$ be the binary matrix of size $(w \times (2^w - 1))$ that has in column number $i, i \in [0, 2^w - 2]$ the vector $(\epsilon^i)_w$. For each $i \in [0, 2^w - 2]$, let $\mathcal{U}_i$ be the matrix which is obtained from $\mathcal{U}_0$ by cyclically rotating its columns i places to the left. Note that for each $i \in [0, 2^w - 2]$ the first column in matrix $\mathcal{U}_i$ is the vector $(\epsilon^i)_w$.

For an element ϵ^i over $\mathbb{F}_{2^w}$ let $\mathcal{T}(\epsilon^i) = \mathcal{U}_i$ be a matrix over Σ of size $(w \times (2^w - 1))$ and let $\mathcal{T}(0)$ be the $(w \times (2^w - 1))$ zeros matrix. We define the same transformation for vectors and matrices, where for a matrix M_1 of size $(a \times b)$ over $\mathbb{F}_{2^w}$ let $\mathcal{T}(M_1) = M_2$ be the matrix over Σ of size $(aw \times b(2^w - 1))$ that is obtained from M_1 by replacing each element α of $\mathbb{F}_{2^w}$ in the matrix M_1 by its appropriate $(w \times (2^w - 1))$ matrix $\mathcal{T}(\alpha)$.

The following is an example to demonstrate Definition 47.

Example 6: Let $\mathcal{B} = \{1, \epsilon^1, \epsilon^2\}$ be a basis for $\mathbb{F}_{2^3}$ over Σ, where ϵ is a primitive element of $\mathbb{F}_{2^3}$ chosen to satisfy the primitive polynomial $x^3 + x + 1$, and hence, $\epsilon^3 = \epsilon + 1$. Then, the coordinates of the successive powers of ϵ with respect to $\mathcal{B}$ are the columns of the matrix $\mathcal{U}_0$

$$\mathcal{U}_0 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

For example, the following matrix is $\mathcal{T}(\epsilon^1)$

$$\mathcal{T}(\epsilon^1) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

We show that the transformation defined in Definition 47 is a linear transformation.

Lemma 48: The transformation $\mathcal{T} : \mathbb{F}_{2^w} \to \mathbb{F}_2^{w \times (2^w - 1)}$ is a linear transformation.

Proof: We want to show that for any $\epsilon^{i_1}, \epsilon^{i_2} \in \mathbb{F}_{2^w}$, $\mathcal{T}(\epsilon^{i_1}) + \mathcal{T}(\epsilon^{i_2}) = \mathcal{T}(\epsilon^{i_1} + \epsilon^{i_2})$. Assume that $\epsilon^{i_1} + \epsilon^{i_2} = \epsilon^{i_3}$. From Definition 47, we know that $\mathcal{T}(\epsilon^{i_1}) + \mathcal{T}(\epsilon^{i_2}) = \mathcal{U}_{i_1} + \mathcal{U}_{i_2}$. From Definition 47, for every $j \in [2^w - 1]$, the j-th column of $\mathcal{U}_{i_1}, \mathcal{U}_{i_2}, \mathcal{U}_{i_3}, \mathcal{U}_{i_1} + \mathcal{U}_{i_2}$ is $(\epsilon^{i_1+j})_w, (\epsilon^{i_2+j})_w, (\epsilon^{i_3+j})_w, (\epsilon^{i_1+j} + \epsilon^{i_2+j})_w$, respectively. Also, $\epsilon^{i_1+j} + \epsilon^{i_2+j} = \epsilon^j(\epsilon^{i_1} + \epsilon^{i_2}) = \epsilon^{i_3+j}$. Thus, the j-th column of $\mathcal{U}_{i_3}$ is equal to the j-th column of $\mathcal{U}_{i_1} + \mathcal{U}_{i_2}$ for all $j \in [2^w - 1]$. Thus, $\mathcal{T}(\epsilon^{i_1}) + \mathcal{T}(\epsilon^{i_2}) = \mathcal{U}_{i_1} + \mathcal{U}_{i_2} = \mathcal{U}_{i_3} = \mathcal{T}(\epsilon^{i_1} + \epsilon^{i_2})$. ∎

The same transformation $\mathcal{T}$ that was defined for vectors and matrices in Definition 47 is also a linear transformation following similar proof as for Lemma 48. The following result can be found in [7, Lemma 3.1], but we want to prove it in a different way, by constructing a specific parity check matrix in order to use it in other claims.

Lemma 49: Let H be a parity check matrix of an $[n, n-s, r]_{2^w}$ covering code. Then, the matrix $\mathcal{T}(H)$ is a parity check matrix of a binary $[(2^w - 1)n, (2^w - 1)n - ws, r]$ covering code. In particular, $h[ws, r] \leqslant (2^w - 1) \cdot h[s, r]_{2^w}$.

Proof: Let $\mathcal{C}$ be an $[n, n-s, r]_{2^w}$ covering code and let H be a parity check matrix of the code $\mathcal{C}$ of size $(s \times n)$. We want to show that the matrix $H' = \mathcal{T}(H)$ is a parity check matrix of a binary $[(2^w - 1)n, (2^w - 1)n - ws, r]$ covering code. The size of H' is $(ws \times (2^w - 1)n)$. Given a binary column vector $\boldsymbol{u}$ of length ws, we show that there are at most r columns of H' that their sum is $\boldsymbol{u}$.

The vector $\boldsymbol{u}$ can be partitioned into s vectors of length w where $\boldsymbol{u} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_s)^\intercal$. Each vector $\boldsymbol{u}_i$ of length w can represent an element of $\mathbb{F}_{2^w}$ according to the basis $\mathcal{B}$ from Definition 47. Hence, $\boldsymbol{u} = ((\epsilon^{i_1})_w^\intercal, (\epsilon^{i_2})_w^\intercal, \ldots, (\epsilon^{i_s})_w^\intercal)^\intercal$ and from the s elements we can get a column vector $\boldsymbol{v} = (\epsilon^{i_1}, \epsilon^{i_2}, \ldots, \epsilon^{i_s})^\intercal$ of dimension $s \times 1$ over $\mathbb{F}_{2^w}$. The first column in each $\mathcal{U}_i, i \in [0, 2^w - 2]$ is the vector $(\epsilon^i)_w$. Then, from the construction of $\mathcal{T}(\boldsymbol{v})$, the first column of the matrix $\mathcal{T}(\boldsymbol{v})$ is the vector $\boldsymbol{u}$.

From the property of the code $\mathcal{C}$, it is known that there exists a vector $\boldsymbol{y} \in \mathbb{F}_{2^w}^n$ such that $H \cdot \boldsymbol{y} = \boldsymbol{v}$, where $w_H(\boldsymbol{y}) \leqslant r$. Let $\mathcal{A} = \{i : i \in [n], y_i \neq 0\}$ and note that $|\mathcal{A}| \leqslant r$. Let $\boldsymbol{h}_i$ be the i-th column of H. Then, $\sum_{i \in \mathcal{A}} y_i \boldsymbol{h}_i = \boldsymbol{v}$. For each $i \in \mathcal{A}$ we define $\boldsymbol{h}'_i = y_i \boldsymbol{h}_i$ and from the linearity of the transformation $\mathcal{T}$ we have $\mathcal{T}(\boldsymbol{v}) = \mathcal{T}(\sum_{i \in \mathcal{A}} \boldsymbol{h}'_i) = \sum_{i \in \mathcal{A}} \mathcal{T}(\boldsymbol{h}'_i)$. Thus, the vector $(\sum_{i \in \mathcal{A}} \mathcal{T}(\boldsymbol{h}'_i))_1 = \sum_{i \in \mathcal{A}} \mathcal{T}(\boldsymbol{h}'_i)_1 = \boldsymbol{u}$, where $\mathcal{T}(\boldsymbol{h}'_i)_1$ is the first column of the matrix $\mathcal{T}(\boldsymbol{h}'_i)$.

For each $i \in \mathcal{A}$, assume that $y_i = \epsilon^{j_i}$. Then, the first column of the matrix $\mathcal{T}(\boldsymbol{h}'_i)$ is the j_i-th column of the matrix $\mathcal{T}(\boldsymbol{h}_i)$. Thus, $\sum_{i \in \mathcal{A}} \mathcal{T}(\boldsymbol{h}_i)_{j_i} = \boldsymbol{u}$, where $\mathcal{T}(\boldsymbol{h}_i)_{j_i}$ is the j_i-th column of the matrix $\mathcal{T}(\boldsymbol{h}_i)$. For each $i \in \mathcal{A}$, the matrix $\mathcal{T}(\boldsymbol{h}_i)$ has size $(ws \times (2^w - 1))$ and it is a sub matrix of H' that starts in the column number $(2^w - 1)(i-1)+1$ of H'. Hence, the j_i-th column of the matrix $\mathcal{T}(\boldsymbol{h}_i)$ is the column number $(2^w - 1)(i-1) + j_i$ of the matrix H'. Therefore, $\sum_{i \in \mathcal{A}} \boldsymbol{h}'_{(2^w-1)(i-1)+j_i} = \boldsymbol{u}$, where $\boldsymbol{h}'_i$ is the i-th column of H'. Thus, the vector $\boldsymbol{u}$

is a sum of $|\mathcal{A}| \leqslant r$ columns of H' and the matrix H' is a parity check matrix of a binary $[(2^w-1)n, (2^w-1)n-ws, r]$ covering code. ∎

An upper bound on the value of $D(s,1,t,r)$ can be obtained in the next theorem using non-binary covering codes.

Theorem 50: For any positive integer w such that $t|w$, $D(ws,1,t,r) \leqslant \dfrac{(2^w-1)h[s,r]_{2^w}}{2^t-1}$.

Proof: Let $\mathcal{C}$ be an $[n, n-s, r]_{2^w}$ covering code over $\mathbb{F}_{2^w}$, where $n = h[s,r]_{2^w}$. Let the matrix H be a parity check matrix of $\mathcal{C}$ of size $(s \times n)$. From Lemma 49, we get that there exists a binary $[(2^w-1)n, (2^w-1)n-ws, r]$ covering code with parity check matrix $H' = \mathcal{T}(H)$. We want to find the smallest size of a t-partition of H'.

Let $j_1, j_2, j_3 \in [0, 2^w-2]$ be such that $\epsilon^{j_1} + \epsilon^{j_2} = \epsilon^{j_3}$. Then, in the matrix $\mathcal{U}_0$ from Definition 47, it holds that the sum of the j_1-th and j_2-th columns is the j_3-th column. In the matrix $\mathcal{U}_i, i \in [0, 2^w-2]$ the j_1-th, j_2-th, j_3-th column is $(\epsilon^i \cdot \epsilon^{j_1})_w, (\epsilon^i \cdot \epsilon^{j_2})_w, (\epsilon^i \cdot \epsilon^{j_3})_w$, respectively. It holds that $\epsilon^i \cdot \epsilon^{j_1} + \epsilon^i \cdot \epsilon^{j_2} = \epsilon^i \cdot (\epsilon^{j_1} + \epsilon^{j_2}) = \epsilon^i \cdot \epsilon^{j_3}$. Thus, we can conclude that in the matrix $\mathcal{U}_i, i \in [0, 2^w-2]$ it also holds that the sum of the j_1-th and j_2-th columns is the j_3-th column. Let $(\mathcal{U}_i)_j$ be the j-th column of $\mathcal{U}_i$. Assume that a basis that includes the columns $\{(\mathcal{U}_0)_{j_1}, (\mathcal{U}_0)_{j_2}, \ldots, (\mathcal{U}_0)_{j_t}\}$ spans the columns $\{(\mathcal{U}_0)_{j_1}, (\mathcal{U}_0)_{j_2}, \ldots, (\mathcal{U}_0)_{j_{2^t-1}}\}$ of the matrix $\mathcal{U}_0$. Then, the basis that includes the columns $\{(\mathcal{U}_i)_{j_1}, (\mathcal{U}_i)_{j_2}, \ldots, (\mathcal{U}_i)_{j_t}\}$ spans the columns $\{(\mathcal{U}_i)_{j_1}, (\mathcal{U}_i)_{j_2}, \ldots, (\mathcal{U}_i)_{j_{2^t-1}}\}$ of the matrix $\mathcal{U}_i, i \in [0, 2^w-2]$.

The matrix $\mathcal{U}_0$ includes all the nonzero column vectors of length w, which means that it includes the space $\mathbb{F}_2^w \setminus \{0\}$. It is given that $t|w$. Hence, there exists a t-spread of $\mathbb{F}_2^w$. Thus, there exists a strict t-partition $\mathcal{P}$ of $\mathcal{U}_0$ with $p = \frac{2^w-1}{2^t-1}$ t-subspaces. Each subspace of $\mathcal{P}$ is represented by a basis of t column vectors of $\mathcal{U}_0$ and denote them by $\{\{(\mathcal{U}_0)_{j_1^1}, (\mathcal{U}_0)_{j_2^1}, \ldots, (\mathcal{U}_0)_{j_t^1}\}, \{(\mathcal{U}_0)_{j_1^2}, (\mathcal{U}_0)_{j_2^2}, \ldots, (\mathcal{U}_0)_{j_t^2}\}, \ldots, \{(\mathcal{U}_0)_{j_1^p}, (\mathcal{U}_0)_{j_2^p}, \ldots, (\mathcal{U}_0)_{j_t^p}\}\}$. The p t-subspaces $\{\{(\mathcal{U}_i)_{j_1^1}, (\mathcal{U}_i)_{j_2^1}, \ldots, (\mathcal{U}_i)_{j_t^1}\}, \{(\mathcal{U}_i)_{j_1^2}, (\mathcal{U}_i)_{j_2^2}, \ldots, (\mathcal{U}_i)_{j_t^2}\}, \ldots, \{(\mathcal{U}_i)_{j_1^p}, (\mathcal{U}_i)_{j_2^p}, \ldots, (\mathcal{U}_i)_{j_t^p}\}\}$ form a strict t-partition of $\mathcal{U}_i$. For each $i \in [n]$ let $\boldsymbol{h}_i$ be the i-th column of the matrix H. The matrix $\mathcal{T}(\boldsymbol{h}_i)$ includes s matrices of size $(w \times (2^w-1))$ that all have the same partition regarding the column numbers. Hence, the partition $\{\{((\mathcal{U}_{i_1})^{\intercal}_{j_1^1}, (\mathcal{U}_{i_2})^{\intercal}_{j_1^1}, \ldots, (\mathcal{U}_{i_s})^{\intercal}_{j_1^1})^{\intercal}, \ldots, ((\mathcal{U}_{i_1})^{\intercal}_{j_t^1}, (\mathcal{U}_{i_2})^{\intercal}_{j_t^1}, \ldots, (\mathcal{U}_{i_s})^{\intercal}_{j_t^1})^{\intercal}\}, \ldots, \{((\mathcal{U}_{i_1})^{\intercal}_{j_1^p}, (\mathcal{U}_{i_2})^{\intercal}_{j_1^p}, \ldots, (\mathcal{U}_{i_s})^{\intercal}_{j_1^p})^{\intercal}, \ldots, ((\mathcal{U}_{i_1})^{\intercal}_{j_t^p}, (\mathcal{U}_{i_2})^{\intercal}_{j_t^p}, \ldots, (\mathcal{U}_{i_s})^{\intercal}_{j_t^p})^{\intercal}\}\}$ is a strict t-partition of $\mathcal{T}(\boldsymbol{h}_i)$ with $p = \frac{2^w-1}{2^t-1}$ t-subspaces. Therefore, there exits a strict t-partition of the matrix H' with $\frac{(2^w-1)n}{2^t-1}$ t-subspaces. Thus, By using Theorem 46 we get that $D(ws,1,t,r) \leqslant \dfrac{(2^w-1)h[s,r]_{2^w}}{2^t-1}$. ∎

We can use Theorem 50 to find upper bounds on the value of $D(s,1,t,r)$ by using previous bounds on the size of non-binary covering codes.

Example 7:

a) In [14] a $[1097, 1097-8, 2]_{2^3}$ covering code is provided. Thus, $h[8,2]_{2^3} \leqslant 1097$. Then, from Theorem 50, $D(3 \cdot 8,1,3,2) = D(24,1,3,2) \leqslant \frac{2^3-1}{2^3-1} h[8,2]_{2^3} = 1097$. For a lower bound, we can use Theorem 38(a) to get $D(24,1,3,2) \geqslant 828$.

b) For $r=3$, the following result can be obtained from [13, Theorem 4.3]. For $q=4$ and $p=3$, $h[s=3p+2,3]_q \leqslant (9 \cdot q^2 + 2\frac{q^2-1}{q-1}) = 154$. Hence, $h[11,3]_{2^2} \leqslant 154$. From Theorem 50, $D(22,1,2,3) \leqslant 154$. For a lower bound, we can use Theorem 38(a) to get $D(22,1,2,3) \geqslant 99$.

The following is another use of Theorem 50 to find bounds on the value of $D(s,1,t,r)$ using another general family of non-binary covering codes.

Corollary 51: For any positive integers w and t, where $t|w$, $D(4w,1,t,2) \leqslant \dfrac{(2^w-1)(2^{w+1}+1)}{2^t-1}$.

Proof: In [7, Theorem 3.2] there exists a construction of a $(4 \times (2^{w+1}+1))$ parity check matrix H of a $[2^{w+1}+1, 2^{w+1}+1-4, 2]_{2^w}$ covering code over $\mathbb{F}_{2^w}$. Therefore, $h[4,2]_{2^w} \leqslant 2^{w+1}+1$. From Theorem 50 we get $D(4w,1,t,2) \leqslant \dfrac{(2^w-1)(2^{w+1}+1)}{2^t-1}$. ∎

For any positive integers w and t, where $t|w$ we have $D(4w,1,t,2) \leqslant 2 \cdot \frac{2^{2w}-1}{2^t-1}$ from Theorem 43(b), and from Corollary 51 we get $D(4w,1,t,2) \leqslant \frac{(2^w-1)(2^{w+1}+1)}{2^t-1}$. Thus, we can save $2 \cdot \frac{2^{2w}-1}{2^t-1} - \frac{(2^w-1)(2^{w+1}+1)}{2^t-1} = \frac{2^w-1}{2^t-1}$ buckets.

The following is an example of a locality functional array code that is obtained from Corollary 51.

Example 8: For the case of $w=4$ and $t=2$, we have $s = 4w = 16$. Let V be $\mathbb{F}_{2^w}^4 = \mathbb{F}_{16}^4$. To get a basis for V as a vector space over Σ, we first choose a basis $\mathcal{B} = \{1, \epsilon, \epsilon^2, \epsilon^3\}$ for $\mathbb{F}_{16}$ over Σ where ϵ is a primitive element of $\mathbb{F}_{16}$ chosen to satisfy the primitive polynomial x^4+x+1. It holds that $\epsilon^4 = \epsilon + 1$. Then, the coordinates of the successive powers of ϵ with respect to the basis $\mathcal{B}$ are the columns of the matrix

$$\mathcal{U}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

In [7, Theorem 3.2], there exists a construction of a parity check matrix of a $[33, 33-4, 2]_{2^w}$ covering code.

$$H = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & \epsilon^1 & \epsilon^2 & \cdots & \epsilon^{14} & 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & \epsilon^2 & \epsilon^4 & \cdots & \epsilon^{13} & 0 & 0 & 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 & 1 & \epsilon^1 & \cdots & \epsilon^{14} \end{bmatrix}.$$

Let $(\mathcal{U}_i)_j$ be the j-th column of $\mathcal{U}_i$. The following is a strict t-partition of $\mathcal{U}_i$, $\mathcal{P}_i = \{\{(\mathcal{U}_i)_1, (\mathcal{U}_i)_6, (\mathcal{U}_i)_{11}\}, \{(\mathcal{U}_i)_2, (\mathcal{U}_i)_7, (\mathcal{U}_i)_{12}\}, \{(\mathcal{U}_i)_3, (\mathcal{U}_i)_8, (\mathcal{U}_i)_{13}\}, \{(\mathcal{U}_i)_4, (\mathcal{U}_i)_9, (\mathcal{U}_i)_{14}\}, \{(\mathcal{U}_i)_5, (\mathcal{U}_i)_{10}, (\mathcal{U}_i)_{15}\}\}$, where we represent every subspace in $\mathcal{P}$ by its elements except the zero vector. In addition, each subspace can be represented by a basis of two vectors.

From Lemma 49 we get that $H' = \mathcal{T}(H)$ is a parity check matrix of a binary $[495, 495-16, 2]$ covering code. Recall that in the transformation $\mathcal{T}$, each element of $\mathbb{F}_{16}$ is replaced with an appropriate matrix $\mathcal{U}_i$ of size (4×15). Each column in H has 4 elements of $\mathbb{F}_{16}$ and is replaced with 4 matrices such that each matrix $\mathcal{U}_i$ of size (4×15) that has a strict t-partition $\mathcal{P}_i$ with 5 subspaces. Each column in H is a (16×15) matrix in H', which can be stored in 5 buckets

such that each bucket stores one subspace from the partition, and hence, the 33 columns of H can be stored in $33*5 = 165$ buckets. Thus, we get that $D(16, 1, 2, 2) \leqslant 165$.

Next, another possible way to obtain locality functional array codes from covering codes is presented. First, we define a possible modification for matrices that we will use in order to construct new parity check matrices for covering codes from given parity check matrices.

Definition 52: Given a matrix H of size $(n \times s)$, its i-th **modified matrix** denoted by $H^{(i)}$ of size $(n+1 \times s)$ is the matrix that has the same rows of H except of row i, where it has the complement of row i of H, with an additional column with only 1 in row i.

The next theorem shows that for a given parity check matrix of a covering code, the modified matrix is also a parity check matrix of another covering code. Even though the following seems to be a basic property, we could not find its proof, and hence, we add the following proof for completeness.

Theorem 53: For a parity check matrix H for a binary $[n, n-s, 2]$ covering code and an integer i, the i-th modified matrix $H^{(i)}$ is also a parity check matrix of a binary $[n+1, n+1-s, 2]$ covering code.

Proof: Let H be a parity check matrix of an $[n, n-s, 2]$ covering code. For a given $i \in [s]$, let $H^{(i)}$ be the i-th modified matrix of H. The size of $H^{(i)}$ is $(s \times (n+1))$. From Property 13, for each vector $\boldsymbol{v} \in \Sigma^s$ there exists a vector $\boldsymbol{y} \in \Sigma^n$ such that $H \cdot \boldsymbol{y} = \boldsymbol{v}$ where $w_H(\boldsymbol{y}) \leqslant 2$. Let $\boldsymbol{h}_i, \boldsymbol{h}'_i$ be the i-th column of $H, H^{(i)}$, respectively. If $w_H(\boldsymbol{y}) = 2$, assume that $\boldsymbol{v} = \boldsymbol{h}_{j_1} + \boldsymbol{h}_{j_2}$. The column vector $\boldsymbol{h}'_j$ is different from the column vector $\boldsymbol{h}_j$ only in row i, where $\boldsymbol{h}'_j$ has the complement of the element in row i in $\boldsymbol{h}_j$. Thus, it holds that $\boldsymbol{v} = \boldsymbol{h}'_{j_1} + \boldsymbol{h}'_{j_2}$.

If $w_H(\boldsymbol{y}) = 1$, assume that $\boldsymbol{v} = \boldsymbol{h}_j$. From the construction of $H^{(i)}$, it holds that $\boldsymbol{h}_j = \boldsymbol{h}'_j + \boldsymbol{h}'_{n+1}$. Therefore, we can get $\boldsymbol{v}$ as a sum of two columns of $H^{(i)}$. Thus, $H^{(i)}$ is a parity check matrix of a binary $[n+1, n+1-s, 2]$ covering code. ∎

One possible way to use Theorem 53 to get locality functional array codes is shown next.

Theorem 54: $D(7, 1, 2, 2) = 7$.

Proof: From [20, Theorem 1] and the example after it, we can get a construction of a parity check matrix for a binary $[19, 19-7, 2]$ covering code. The following is a parity check matrix H of the code.

$$\begin{bmatrix}
0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&0&0\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&0&0&0\\
0&0&0&0&0&1&1&0&1&0&1&0&1&1&0&0&0&1&1\\
0&0&0&0&1&1&0&0&0&1&1&0&1&0&1&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1
\end{bmatrix}.$$

The following is the matrix $H^{(1)}$, the first modified matrix of H where the first row is the complement of the first row of H and a new column with only 1 in the first entry is added.

$$\begin{bmatrix}
1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&0&0\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&0&0&0\\
0&0&0&0&0&1&1&0&1&0&1&0&1&1&0&0&0&1&0\\
0&0&0&0&1&1&0&0&0&1&1&0&1&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&0
\end{bmatrix}.$$

From Theorem 53, the matrix $H^{(1)}$ is a parity check matrix of a binary $[20, 20-7, 2]$ covering code. Note that the fourth column is all zero column which we can remove to get the following matrix $H^{(1)'}$ which is a parity check matrix of a binary $[19, 19-7, 2]$ covering code.

$$\begin{bmatrix}
1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&1&1&0&1&1&0&0&1&1&0&0&1&1&0&0&0&0&0\\
1&0&1&1&0&1&0&1&0&1&0&1&0&1&0&0&0&0&0\\
0&0&0&0&1&1&0&1&0&1&0&1&1&0&0&0&1&1&0\\
0&0&0&1&1&0&0&0&1&1&0&1&0&1&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0\\
0&0&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0
\end{bmatrix}.$$

Let $\boldsymbol{h}'_j$ be the j-th column of the matrix $H^{(i)'}$. We can find a 2-partition of the matrix $H^{(1)'}$. We will present the partition as a set of 7 2-subspaces such that each subspace is presented by a basis with two columns of $H^{(i)'}$. The following is a possible 2-partition of $H^{(i)'}$ $\mathcal{P} = \{\{\boldsymbol{h}'_7, \boldsymbol{h}'_{11}\}, \{\boldsymbol{h}'_8, \boldsymbol{h}'_{12}\}, \{\boldsymbol{h}'_9, \boldsymbol{h}'_{13}\}, \{\boldsymbol{h}'_{10}, \boldsymbol{h}'_{14}\}, \{\boldsymbol{h}'_4, \boldsymbol{h}'_5\}, \{\boldsymbol{h}'_1, \boldsymbol{h}'_2\}, \{\boldsymbol{h}'_3, \boldsymbol{h}'_{19}\}\}$. We can see that 14 out of 19 columns form the bases. It can be verified that $\boldsymbol{h}'_4 + \boldsymbol{h}'_5 = \boldsymbol{h}'_6$, $\boldsymbol{h}'_7 + \boldsymbol{h}'_{11} = \boldsymbol{h}'_{15}$, $\boldsymbol{h}'_8 + \boldsymbol{h}'_{12} = \boldsymbol{h}'_{16}$, $\boldsymbol{h}'_{10} + \boldsymbol{h}'_{14} = \boldsymbol{h}'_{17}$ and $\boldsymbol{h}'_9 + \boldsymbol{h}'_{13} = \boldsymbol{h}'_{18}$. Therefore, $\mathcal{P}$ is a 2-partition of $H^{(i)'}$ with size 7. Thus, from Theorem 46 we get that $D(7, 1, 2, 2) \leqslant 7$.

For the lower bound, assume by contradiction that there exists a $(7, 1, 6, 2, 2)$ locality functional array code. Then, from Theorem 55 we get that $h[7, 2] \leqslant 18$. But from [12] we have that $h[7, 2] = 19$, which is a contradiction. Thus, $D(7, 1, 2, 2) \geqslant 7$. ∎

Next, we show how to construct covering codes using locality functional array codes.

Theorem 55: Let $\mathcal{C}$ be an $(s, 1, m, t, r)$ locality functional array code. Then, $h[s, r] \leqslant m \cdot (2^t - 1)$.

Proof: Assume that $\mathcal{C}$ is an $(s, 1, m, t, r)$ locality functional array code which has m buckets such that in each bucket stored at most t linear combinations of the s information bits. From the t cells in each bucket we can get at most $(2^t - 1)$ different linear combinations. We can represent each linear combination as a binary vector of length s. Then, we construct an $(s \times m \cdot (2^t - 1))$ parity check matrix H where we have all the vectors that we get from the linear combinations of all the m buckets as columns of the matrix. Let $\boldsymbol{u} \in \Sigma^s$ be a column vector of length s which can represent a request for the code $\mathcal{C}$. From the property of $\mathcal{C}$, there exists a recovering set $S \subseteq [m]$ where $|S| \leqslant r$ that satisfies the request. Assume that $S = \{b_1, b_2, \ldots, b_{r'}\}$ where $r' \leqslant r$. From each bucket $b_i \in S$ we read a linear combination $\boldsymbol{v}_i$ of the t cells which is a linear combination of the s information bits. From the construction of H, the column vector $\boldsymbol{v}_i$ is a column in H. Then, $\boldsymbol{u} = \sum_{i=1}^{r'} \boldsymbol{v}_i$, and hence, the vector $\boldsymbol{u}$ is a sum of at

TABLE XI
SUMMARY OF THE RESULTS

Code	s	k	t	ℓ	r	Lower bound	Upper bound	Notes
FP/FB	s	1	t	t	$-$	$\left\lceil \frac{s}{t} \right\rceil$	$\left\lceil \frac{s}{t} \right\rceil$	Theorem 16
FP/FB	s	1	t	1	$-$	$\left\lceil \frac{s}{\log_2(t+1)} \right\rceil$	$\left\lceil \frac{s}{\lfloor \log_2(t+1) \rfloor} \right\rceil$	Theorem 16
FP/FB	s	1	t	$t/2$	$-$	$\frac{s}{t}+1$	$\frac{s}{t}+1$	t is even, $\frac{s}{t}$ is integer, and $\frac{s}{t} \leqslant t-1$ Theorem 16
FP/FB	s_1+s_2	1	t	1	$-$	$\left\lceil \frac{s_1+s_2}{\log_2(t+1)} \right\rceil$	$\left\lceil \frac{s_1}{\lfloor \log_2(t+1) \rfloor} \right\rceil + 1$	$2^{s_2}-1 \leqslant \left(\left\lceil \frac{s_1}{\lfloor \log_2(t+1) \rfloor} \right\rceil + 1 \right) \cdot (t-(2^{\lfloor \log_2(t+1) \rfloor}-1))$ Theorem 17
FP/FB	s	1	t	αt	$-$		$\left\lceil \frac{s}{t-g\lceil t,\alpha t \rceil} \right\rceil$	$0<\alpha<1$ Theorem 16
FB	s	k	t	1	$-$		$FB(\frac{s}{t}, t \cdot k)$	$\frac{s}{t}$ is integer Lemma 10
FB	8	2	2	2	$-$	6	7	Theorem 18
FB	s	2	2	2	$-$	$\log_7(2^{s-1} \cdot (2^s-1))$	$7 \cdot \left\lceil \frac{s}{8} \right\rceil$	Corollary 19
P	r^2+r	r	r^2-r+1	$r-1$	$-$	$r+1$	$r+1$	$r \geqslant 3$ Theorem 28
B	r^2+r	r	r^2-r+1	$r-1$	$-$	$r+1$	$r+1$	$r \geqslant 3$ Theorem 30
B	6	15	2	2	$-$	25	25	Theorem 25
FP	6	11	2	2	$-$	21	25	Theorem 26
P	4	16	2	1	$-$	23	25	Theorem 32
FP	4	14	2	2	$-$	24	25	Theorem 33
FP	5	48	2	2	$-$	88	90	Theorem 34
L	s	1	t	t	1	$\left\lceil \frac{2^s-1}{2^t-1} \right\rceil$	$\left\lceil \frac{2^s-1}{2^t-1} \right\rceil$	Theorem 43
L	s	1	t	t	r		$r \cdot \left\lceil \frac{2^{s/r}-1}{2^t-1} \right\rceil$	$r \mid s$ Theorem 43
L	s	$\begin{bmatrix} s-1 \\ t-1 \end{bmatrix}_2$	t	t	1		$\begin{bmatrix} s \\ t \end{bmatrix}_2$	Theorem 43
L	s	$\left\lfloor \frac{2^s-2^t}{r \cdot 2^t-r} \right\rfloor + 1$	t	t	r		$\frac{2^s-1}{2^t-1}$	$s=rt$ Theorem 43
L	3	2	2	2	1	5	6	Example 4
L	s	1	1	1	r	$h[s,r]$	$h[s,r]$	Theorem 44
L	ws	1	t	t	r	$\left\lceil \frac{h[ws,r]}{2^t-1} \right\rceil$	$\frac{(2^w-1)h[s,r]2^w}{2^t-1}$	$t \mid w$ Corollary 56, Theorem 50
L	$4w$	1	t	t	2		$\frac{(2^w-1)(2^{w+1}+1)}{2^t-1}$	$t \mid w$ Corollary 51
L	24	1	3	3	2	828	1097	Example 7
L	22	1	2	2	3	99	154	Example 7
L	7	1	2	2	2	7	7	Theorem 54

most r columns of H. Thus, the matrix H is a parity check matrix of a binary $[m \cdot (2^t-1), m \cdot (2^t-1)-s, r]$ covering code, and hence, $h[s,r] \leqslant m \cdot (2^t-1)$. ■

Now we will use Theorem 55 to get a lower bound on the value of $D(s,1,t,r)$.

Corollary 56: $D(s,1,t,r) \geqslant \left\lceil \dfrac{h[s,r]}{2^t-1} \right\rceil$.

Proof: Assume by contradiction that $D(s,1,t,r) = m < \left\lceil \dfrac{h[s,r]}{2^t-1} \right\rceil$. The number of buckets m is an integer. Then, $m < \dfrac{h[s,r]}{2^t-1}$. From Theorem 55 we have $h[s,r] \leqslant m \cdot (2^t-1) < \dfrac{h[s,r]}{2^t-1} \cdot (2^t-1) = h[s,r]$ which is a contradiction. ■

We can get upper bounds on the value $h[s,r]$ from [12]. For example, $h[2s-1,2] \geqslant 2^s - 1$ for any $s \geqslant 3$ and we can conclude that $D(2s-1,1,t,2) \geqslant \left\lceil \frac{2^s-1}{2^t-1} \right\rceil$.

IX. Conclusion

In this work we studied constructions and bounds of several families of codes. We defined and presented functional PIR array codes, functional batch array codes, and locality functional array codes. Lower bounds on the smallest number of buckets of these codes were given. Several upper bounds on the smallest number of buckets were shown based on general constructions, specific constructions, subspaces, and covering codes. In Table XI, we provide a summary of most of the results that appear in the work. The first column specifies the family of codes that the result refers to. Denote a PIR array code, batch array code, functional PIR array code, functional batch array code, locality functional array code by P, B, FP, FB, L, respectively. The next five columns specify the values of the parameters of the codes. The following two columns refer to lower and upper bounds on the codes and the last column includes notes such as constraints on the parameters and where the results appeared in the work. Lastly, we note that there are plenty of problems which remain for future research, such as generalizing the specific constructions and finding new bounds for different parameters.

References

[1] H. Asi and E. Yaakobi, "Nearly optimal constructions of PIR and batch codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 151–155.

[2] 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.

[3] 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 Symp. Found. Comput. Sci.*, Vancouver, BC, Canada, 2002, pp. 261–270.

[4] S. R. Blackburn and T. Etzion, "PIR array codes with optimal PIR rates," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017. pp. 2658–2662.

[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] S. Buzaglo, Y. Cassuto, P. H. Siegel, and E. Yaakobi, "Consecutive switch codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2485–2498, Apr. 2018.

[7] R. A. Brualdi, V. S. Pless, and R. M. Wilson, "Short codes with a given covering radius," *IEEE Trans. Inf. Theory*, vol. 35, no. 1, pp. 99–109, Jan. 1989.

[8] T. H. Chan, S.-W. Ho, and H. Yamamoto, "Private information retrieval for coded storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, Nov. 2015, pp. 2842–2846.

[9] Y. M. Chee, H. M. Kiah, E. Yaakobi, and H. Zhang, "A generalization of Blackburn-Etzion construction for PIR array codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Paris, France, Jul. 2019, pp. 1062–1066.

[10] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, 1998.

[11] Y. M. Chee, F. Gao, S. T. H. Teo, and H. Zhang, "Combinatorial systematic switch codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, Jun. 2015, pp. 241–245.

[12] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein, *Covering Codes*. Amsterdam, The Netherlands: North-Holland, 1997.

[13] A. A. Davydov, M. Giulietti, S. Marcugini, and F. Pambianco, "Linear covering codes of radius 2 and 3," in *Proc. Workshop Coding Theory Days*, St. Petersburg, Russia, Oct. 2008, pp. 12–17.

[14] A. A. Davydov and P. R. J. Ostergard, "Linear codes with covering radius R = 2, 3 and codimension tR," *IEEE Trans. Inf. Theory*, vol. 47, no. 1, pp. 416–421, Jan. 2001.

[15] Z. Dvir and S. Gopi, "2-server PIR with subpolynomial communication," *J. ACM*, vol. 63, no. 4, pp. 1–15, 2016.

[16] S. El-Zanati, G. Seelinger, P. Sissokho, L. Spence, and C. V. Eynden, "On λ-fold partitions of finite vector spaces and duality," *Discrete Math.*, vol. 311, no. 4, pp. 307–318, 2011.

[17] T. Etzion and A. Vardy, "On q-analogs of Steiner systems and covering designs," *Adv. Math. Commun.*, vol. 5, no. 2, pp. 161–176, 2011.

[18] T. Etzion and H. Zhang, "Grassmannian codes with new distance measures for network coding," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4131–4142, Jul. 2019.

[19] A. Fazeli, A. Vardy, and E. Yaakobi, "PIR with low storage overhead: Coding instead of replication," 2015, *arXiv:1505.06241*.

[20] E. M. Gabidulin, A. A. Davydov, and L. M. Tombak, "Linear codes with covering radius 2 and other new covering codes," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 219–224, 1991.

[21] M. Greferath, M. Pavčević, N. Silberstein, and A. Vázquez-Castro, Eds., *Network Coding and Subspace Designs*. Springer, 2017.

[22] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," in *Proc. 26th Annu. ACM Symp. Theory Comput.*, Chicago, IL, USA, 2004, pp. 262–271.

[23] S. Kumar, H.-Y. Lin, E. Rosnes, and A. G. T. 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.

[24] S. Lin and D. J. Costello, *Error Control Coding*. Upper Saddle River, NJ, USA: Prentice-Hall, 2004.

[25] H.-Y. Lin, S. Kumar, E. Rosnes, A. G. I. Amat, and E. Yaakobi, "Weak private information retrieval," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 1257–1261.

[26] H.-Y. Lin and E. Rosnes, "Lengthening and extending binary private information retrieval codes," 2017, *arXiv:1707.03495*.

[27] H. Lipmaa and V. Skachek, "Linear batch codes," in *Coding Theory and Applications* (CIM Series in Mathematical Sciences), vol. 3, 2015, pp. 245–253.

[28] J. L. Massey, *Threshold Decoding*. Cambridge, MA, USA: MIT Press, 1963.

[29] M. Nassar and E. Yaakobi, "Array codes for functional PIR and batch codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 1024–1029.

[30] L. Pamies-Juarez, H. D. L. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Turkey, Jul. 2013, pp. 892–896.

[31] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, and S. Vishwanath, "Locality and availability in distributed storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, HI, USA, Jun. 2014, pp. 681–685.

[32] A. S. Rawat, Z. Song, A. G. Dimakis, and A. Gal, "Batch codes through dense graphs without short cycles," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1592–1604, Apr. 2016.

[33] E. Sharon and I. Alrod, "Coding scheme for optimizing random I/O performance," in *Proc. Non-Volatile Memories Workshop*, San Diego, CA, USA, Apr. 2013, pp. 1–5.

[34] N. Silberstein, T. Etzion, and M. Schwartz, "Locality and availability of array codes constructed from subspaces," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 2648–2660, May 2019.

[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] J. H. Van Lint and R. M. Wilson, *A Course Combinatorics*. Cambridge, U.K.: Cambridge Univ. Press, 1992.
[41] A. Vardy and E. Yaakobi, "Constructions of batch codes with near-optimal redundancy," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 1197–1201.
[42] M. Vajha, V. Ramkumar, and P. V. Kumar, "Binary, shortened projective Reed Muller codes for coded private information retrieval," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 1421–1425.
[43] Z. Wang, H. M. Kiah, Y. Cassuto, and J. Bruck, "Switch codes: Codes for fully parallel reconstruction," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2061–2075, Apr. 2017.
[44] A. Wang, Z. Zhang, and M. Liu, "Achieving arbitrary locality and availability in binary codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, Jun. 2015, pp. 1866–1870.
[45] G. William, "A survey on private information retrieval," *Bull. EATCS*, vol. 82, pp. 72–107, Jan. 2004.
[46] E. Yaakobi and R. Motwani, "Construction of random input-output codes with moderate block lengths," *IEEE Trans. Commun.*, vol. 64, no. 5, pp. 1819–1828, May 2016.
[47] A. Yamawaki, H. Kamabe, and S. Lu, "Construction of parallel RIO codes using coset coding with Hamming codes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Kaohsiung, Taiwan, Nov. 2017, pp. 239–243.
[48] S. Yekhanin, "Private information retrieval," *Commun. ACM*, vol. 53, no. 4, pp. 68–73, 2010.
[49] H. Zhang and V. Skachek, "Bounds for batch codes with restricted query size," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 1192–1196.
[50] 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.
[51] Y. Zhang, T. Etzion, and E. Yaakobi, "Bounds on the length of functional PIR and batch codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4917–4934, Aug. 2020.

Mohammad Nassar (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2018 and 2020, respectively. He has recently joined IBM as a Cloud Researcher. His research interests include information and coding theory with applications to data storage and retrieval.

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, in 2011. From 2011 to 2013, he was a Post-Doctoral 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. He is currently an Associate Professor at the Computer Science Department, Technion—Israel Institute of Technology. His research interests include information and coding theory with applications to non-volatile memories, associative memories, DNA storage, data storage and retrieval, and private information retrieval. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship (2010–2011).