Nearly Optimal Constructions of PIR and Batch Codes

Hilal Asi, *Student Member, IEEE*, and Eitan Yaakobi, *Senior Member, IEEE*

***Abstract*— In this paper, we study two families of codes with availability, namely, *private information retrieval (PIR) codes* and *batch codes*. While the former requires that every information symbol has k mutually disjoint recovering sets, the latter imposes this property for each multiset request of k information symbols. The main problem under this paradigm is to minimize the number of redundancy symbols. We denote this value by $r_P(n,k)$ and $r_B(n,k)$, for PIR codes and batch codes, respectively, where n is the number of information symbols. Previous results showed that for any constant k, $r_P(n,k) = \Theta(\sqrt{n})$ and $r_B(n,k) = \mathcal{O}(\sqrt{n}\log(n))$. In this paper, we study the asymptotic behavior of these codes for non-constant k and specifically for $k = \Theta(n^\epsilon)$. We also study the largest value of k such that the rate of the codes approaches 1 and show that for all $\epsilon < 1$, $r_P(n, n^\epsilon) = o(n)$ and $r_B(n, n^\epsilon) = o(n)$. Furthermore, several more results are proved for the case of fixed k.**

***Index Terms*— PIR codes, batch codes, availability codes, codes with locality.**

I. Introduction

DISTRIBUTED and cloud storage systems today are required to tolerate the failure or unavailability of some of the nodes in the system. The simplest and most commonly used method to accomplish this task is by replication, whereby every node is replicated several times, usually three. This solution has clear advantages due to its simplicity, fast recovery, and efficient *availability*. However, it entails a large storage overhead which becomes costly in large storage systems. Availability in particular plays an important role in supporting high throughput of the system. Consider the case in which a large number of files are stored in a distributed storage system and user requests of the files are received. If each file has one copy or can be recovered only once, and several users request this file, then these requests will have to be served sequentially, which will significantly slow down the system's response time.

In this paper we study two families of codes with availability for distributed storage. The first family of codes, called *private information retrieval* (*PIR*) *codes*, requires that every information symbol has some k mutually disjoint recovering sets. These codes were studied recently in [11] due to their applicability for private information retrieval in a coded storage system [8]. They are also very similar to *one-step majority-logic decodable codes* that were studied a while ago by Massey [19] and later by Lin and Costello [17] and were prompted by applications of error-correction with low-complexity. Since each symbol has several disjoint recovering sets, it can be decoded with low complexity according to the majority values given by all of its recovering sets. The only difference between these two families of codes is that the latter requires that every code symbol, i.e., both the information and redundancy, has a large number of mutually disjoint recovering sets.

The second family of codes, which is a generalization of the first one, was first proposed in the last decade by Ishai *et al.* under the framework of *batch codes* [13]. These codes were originally motivated by different applications such as load-balancing in storage and cryptographic protocols. Under this setting, it is required that every multiset request of k symbols can be recovered by k mutually disjoint recovering sets. Note that every batch code is in particular a PIR code with the same parameters. Here we restrict our definition of batch codes to the so-called *primitive batch codes*, which is a family of coded batch codes that was mostly studied in the literature. A special case of these codes, called *switch codes*, was recently studied for network applications [6], [9], [29]–[31].

Formally, we denote a k-PIR code by $[N,n,k]^P$ to be a coding scheme which encodes n information bits to N bits such that each information bit has k mutually disjoint recovering sets. Similarly, a k-batch code will be denoted by $[N,n,k]^B$ and the requirement of mutually disjoint recovering sets is imposed for every multiset request of size k. The main figure of merit when studying PIR and batch codes is the value of N, given n and k. Thus, we denote by $P(n,k)$, $B(n,k)$ the minimum value of N for which an $[N,n,k]^P$, $[N,n,k]^B$ code exists, respectively. Since it is known that for all fixed k, $\lim_{n\to\infty} B(n,k)/n = \lim_{n\to\infty} P(n,k)/n = 1$, [13], we evaluate these codes by their redundancy and define $r_B(n,k) \triangleq B(n,k) - n$, $r_P(n,k) \triangleq P(n,k) - n$.

It is easy to see that for $k = 2$, $r_B(n,2) = r_P(n,2) = 1$, and for $k \in \{3,4\}$, $r_B(n,k) = r_P(n,k) = \Theta(\sqrt{n})$ [11], [21], [27], [33]. Furthermore, for any other fixed k, $r_P(n,k) = \Theta(\sqrt{n})$ [11], [21], [33] and $r_B(n,k) = O(\sqrt{n}\log(n))$ [27]. One of the problems we study in this paper studies the largest value of k (as a function of n) for which one can still have $r_P(n,k) = o(n)$ and $r_B(n,k) = o(n)$, so the rate of the

Manuscript received August 24, 2017; revised March 26, 2018; accepted June 11, 2018. Date of publication July 2, 2018; date of current version January 18, 2019. This work was supported by the Israel Science Foundation under Grant 1624/14. This paper was presented in part at the 2017 IEEE International Symposium on Information Theory [1].

The authors are with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: shelal@cs.technion.ac.il; yaakobi@cs.technion.ac.il).

Communicated by J. Kliewer, Associate Editor for Coding Techniques.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2018.2852294

TABLE I
BINARY PIR CODES SUMMARY

Construction	Availability k	Redundancy $r_p(n,k)$
Theorem 16	$k = n^\epsilon$, $0 < \epsilon < 1$	$\mathcal{O}(n^{1-\rho})$, for some $\rho > 0$
Theorem 19	$k = n^\epsilon$, $1 \leqslant \epsilon$	$\mathcal{O}(n^{\epsilon+\rho})$ for any $\rho > 0$
Theorem 34	$k \leqslant \sqrt{n}$	$\mathcal{O}(k\sqrt{n})$

TABLE II
BINARY BATCH CODES SUMMARY

Construction	Availability k	Redundancy $r_B(n,k)$
Corollary 28	$k = n^\epsilon$, $0 < \epsilon < 1$	$\mathcal{O}(n^{1-\rho})$, for some $\rho > 0$
Theorem 29	$k = n^\epsilon$, $1 \leqslant \epsilon$	$\mathcal{O}(n^{\epsilon+\rho})$ for any $\rho > 0$
Corollary 48	$k \leqslant \sqrt{n}/\log n$	$\mathcal{O}(k^2\sqrt{n}\log(n))$

codes approaches one. Furthermore, in order to have a better understanding of the asymptotic behavior of the redundancy, we study the values $r_P(n,k)$ and $r_B(n,k)$ when $k = \Theta(n^\epsilon)$.

The results achieved in this paper are based on two constructions. The first one uses *multiplicity codes* which generalize Reed-Muller codes and were first presented by Kopparty *et al.* [15]. These codes were also used for the construction of *locally decodable codes*. The second construction we used is based on the subcube construction from [13]. This basic construction can be used to construct both PIR and batch codes. While the idea in the works in [11] and [13] was to use multidimensional cubes in order to achieve large values of k, here we take a different approach and position the information bits in a two dimensional array and then form multiple parity sets by taking different diagonals in the array. Based on these two constructions and another connection we draw on the existence of batch codes based upon PIR codes, we manage to show that for all $k = \Theta(n^\epsilon)$, where $\epsilon < 1$, $r_P(n,k) = o(n)$ and $r_B(n,k) = o(n)$. Since $r_P(n,k), r_B(n,k) \geqslant k$, this result is indeed optimal. The main results of this work are summarized in Table I and Table II for PIR and batch codes, respectively.

The rest of the paper is organized as follows. In Section II, we formally define the codes studied in this paper and review previous results. In Section III, we review multiplicity codes. Then, in Section IV we show how to use multiplicity codes to construct PIR codes, and in Section V we carry the same task for batch codes. In Section VI, we show how PIR codes can be used to prove the existence of batch codes. Then, in Sections VII and VIII, we present our array construction and its results for PIR and batch codes, respectively. In Section IX, we present another extension of the array construction that is invoked to construct more batch codes. Section X studies the case of $k = 5$ for batch codes and a special case of batch codes in which every bit can be requested at most twice. Finally, Section XI concludes the paper.

II. DEFINITIONS AND PRELIMINARIES

In this section we formally define the codes we study in this paper. Let $\mathbb{F}_q$ denote the field of size q, where q is a prime power. A linear code of length N and dimension n over $\mathbb{F}_q$ will be denoted by $[N,n]_q$. For binary codes we will remove the notation of the field. The set $[n]$ denotes the set of integers $\{1,2,\ldots,n\}$.

In this work we focus on two families of codes, namely *private information retrieval* (*PIR*) codes that were defined recently in [11] and *batch codes* that were first studied by Ishai *et al.* [13]. In these two families of codes, n information symbols are encoded to N symbols. While for PIR codes it is required that every information symbol has k mutually disjoint recovering sets, batch codes impose this property for every multiset request of k symbols. Formally, these codes are defined as follows.

Definition 1: Let $\mathcal{C}$ be an $[N,n]_q$ linear code over the field $\mathbb{F}_q$.

1) *The code $\mathcal{C}$ will be called a k-**PIR code**, and will be denoted by $[N,n,k]_q^P$, if for every information symbol $x_i, i \in [n]$, there exist k mutually disjoint sets $R_{i,1},\ldots,R_{i,k} \subseteq [N]$ such that for all $j \in [k]$, x_i is a function of the symbols in $R_{i,j}$.*
2) *The code $\mathcal{C}$ will be called a k-**batch code**, and will be denoted by $[N,n,k]_q^B$, if for every multiset request of symbols $\{i_1,\ldots,i_k\}$, there exist k mutually disjoint sets $R_{i_1},\ldots,R_{i_k} \subseteq [N]$ such that for all $j \in [k]$, x_{i_j} is a function of the symbols in R_{i_j}.*

PIR codes are similar in their definition to *locally repairable codes* (*LRC*) *with availability* [20], [22], [32], however PIR codes do not impose any constraint on the size of the recovering sets as done for LRCs. In fact, these codes have more in common with *one-step majority-logic decodable codes* that were studied a while ago by Massey [19] and later by Lin and Costello [17] for applications of fast decoding. The main difference is that one-step majority-logic decodable codes require that each symbol (both information and redundancy) will have multiple recovering sets.

We slightly modified here the definition of batch codes. In their conventional definition, n symbols are encoded into some m tuples of strings, called buckets, such that each batch (i.e., request) of k information symbols can be decoded by reading at most some t symbols from each bucket. In case each bucket can store a single symbol, these codes are called *primitive batch codes*, which is the setup we study here and for simplicity call them batch codes. In this work we study the binary and non-binary cases of these codes.

The main problem in studying PIR and batch codes is to minimize the length N given the values of n and k. We denote by $P(n,k)_q, B(n,k)_q$ the value of the smallest N such that there exists an $[N,n,k]_q^P, [N,n,k]_q^B$ code, respectively. Since every batch code is also a PIR code with the same parameters

we get that $B(n,k)_q \geqslant P(n,k)_q$. For the binary case, we will remove q from these and subsequent notations.

Example 2: In case $k = 1$, the code $[n,n]_q$ which simply stores all the information symbols is an $[n,n,1]_q^P$ PIR code and also an $[n,n,1]_q^B$ batch code. Hence $P(n,1)_q = B(n,1)_q = n$. Similarly, the simple parity check code $[n+1,n]_q$ is an $[n+1,n,2]_q^P$ PIR code and also an $[n+1,n,2]_q^B$ batch code, since every symbol has two recovering sets; the symbol itself and all the remaining symbols. Therefore we get that $P(n,2)_q = B(n,2)_q = n+1$.

In [13], it was shown using the subcube construction that for any fixed k there exists an asymptotically optimal construction of $[N,n,k]_q^B$ batch code, and hence

$$\lim_{n\to\infty} B(n,k)_q/n = \lim_{n\to\infty} P(n,k)_q/n = 1.$$

Therefore, it is important to study how fast the rate of these codes converges to one, and so the redundancy of PIR and batch codes is studied. We define $r_B(n,k)_q$ to be the value $r_B(n,k)_q \triangleq B(n,k)_q - n$ and similarly, $r_P(n,k)_q \triangleq P(n,k)_q - n$. Hence, $r_B(n,1)_q = r_P(n,1)_q = 0$, $r_B(n,2)_q = r_P(n,2)_q = 1$.

In [11], it was shown that for any fixed $k \geqslant 3$ it is possible to construct $[N,n,k]$ PIR codes where $N = n + \mathcal{O}(\sqrt{n})$, so $r_P(n,k) = \mathcal{O}(\sqrt{n})$ for any fixed $k \geqslant 3$, and in [21] and [33] it was proved that $r_P(n,3) = \Omega(\sqrt{n})$. Since it is known that $r_B(n,k) \geqslant r_P(n,k) \geqslant r_P(n,3)$ for any fixed $k \geqslant 3$, these results assure also that for any fixed $k \geqslant 3$, $r_P(n,k) = \Theta(\sqrt{n})$ and $r_B(n,k) = \Omega(\sqrt{n})$. In [27], it was proved that for $k = 3,4$, $B(n,k) = P(n,k)$ and hence $r_B(n,k) = \Theta(\sqrt{n})$, and for any fixed $k \geqslant 5$, $r_B(n,k) = \mathcal{O}(\sqrt{n}\log(n))$. In this paper, we will mostly study the values of $r_P(n,k)$ and $r_B(n,k)$, when k is a function of n, for example $k = \Theta(n^\epsilon)$, for $\epsilon > 0$. One of the problems we will also investigate is finding the largest ϵ for which $r_P\big(n, k = \Theta(n^\epsilon)\big) = o(n)$, and similarly for batch codes.

There are several constructions of PIR and batch codes, studied first in [13] and recently in [16], [18], [23], and [28]–[30]. We summarize here the relevant known results for our problem:

1) $r_P(n,k) = \Theta(\sqrt{n})$ for fixed k, [11], [21], [33].
2) $r_P(n,\sqrt{n}) = \mathcal{O}(n^{\frac{\log 3}{2}})$, [17].
3) $r_P(n,n^\epsilon) = \mathcal{O}(n^{0.5+\epsilon})$, [17].
4) $r_P(n,n^{0.25}) = \mathcal{O}(n^{0.714})$, [12].
5) $r_B(n,k) = \mathcal{O}(\sqrt{n}\log n)$ for fixed k, [27].
6) $r_B(n,n^{1/3}) \leqslant n$, [23].
7) $r_B(n,n^\epsilon) \leqslant n^{7/8}$ for $7/32 \leqslant \epsilon \leqslant 1/4$, [23].
8) $r_B(n,n^\epsilon) \leqslant n^{4\epsilon}$ for $1/5 < \epsilon \leqslant 7/32$, [23].
9) $B(n,n) \leqslant 2n^{1.5}$, [6].

Our goal in this work is to close on this gap and study new constructions of PIR and batch codes. Lastly, we note that we only focus on the case where n is large. The case of fixed n or small n comparing to k was studied in [11], [16], and [28] for PIR codes.

Yet another class of related codes is called *combinatorial batch codes*. Here, it is required to have the same properties of batch codes, however the symbols cannot be encoded and thus these codes cannot be compared with the ones studied in this paper. Several works have considered codes under this setup; see e.g. [3], [5], [7], [25].

III. Mutliplicity Codes

In this section we review the construction of *multiplicity codes*. This family of codes was first presented by Kopparty *et al.* [15] as a generalization of Reed-Muller codes by calculating the derivatives of polynomials. These codes inherit the local-decodability property of Reed-Muller codes, and thus were used to construct *locally decodable codes* in [15], but they can achieve higher rates at the same time. We follow the definitions of these codes as were presented in [15] and first start with the definition of the Hasse derivative.

A. Derivatives and Multiplicities

For a field $\mathbb{F}$, let $\mathbb{F}[x_1,x_2,\ldots,x_s] = \mathbb{F}[\boldsymbol{x}]$ be the ring of polynomials in the variables $x_1,x_2,\ldots,x_s$ with coefficients in $\mathbb{F}$. For a vector $\boldsymbol{i} = (i_1,i_2,\ldots,i_s)$ of non-negative integers, its weight $wt(\boldsymbol{i})$ equals $\sum_{j=1}^s i_j$. For a vector of non-negative integers $\boldsymbol{i} = (i_1,i_2,\ldots,i_s)$, let $\boldsymbol{x}^{\boldsymbol{i}}$ denote the monomial $\prod_{j=1}^s x_j^{i_j}$. The total degree of this monomial equals $wt(\boldsymbol{i})$. For $P(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$, let the degree of $P(\boldsymbol{x})$, $\deg(P)$, be the maximum total degree over all monomials in $P(\boldsymbol{x})$.

*Definition 3: For a polynomial $P(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ and a vector $\boldsymbol{i}$ of non-negative integers, the $\boldsymbol{i}$-th **Hasse derivative** of $P(\boldsymbol{x})$, denoted by $P^{(\boldsymbol{i})}(\boldsymbol{x})$, is the coefficient of $\boldsymbol{z}^{\boldsymbol{i}}$ in the polynomial $P'(\boldsymbol{x},\boldsymbol{z}) = P(\boldsymbol{x}+\boldsymbol{z}) \in \mathbb{F}[\boldsymbol{x},\boldsymbol{z}]$.*

It is clear from the definiton that

$$P(\boldsymbol{x}+\boldsymbol{z}) = \sum_{\boldsymbol{i}} P^{(\boldsymbol{i})}(\boldsymbol{x})\boldsymbol{z}^{\boldsymbol{i}}.$$

As for normal derivatives, it also holds that for $P, Q \in \mathbb{F}[\boldsymbol{x}]$ and $\lambda \in \mathbb{F}$,

$$(\lambda P)^{(\boldsymbol{i})}(\boldsymbol{x}) = \lambda P^{(\boldsymbol{i})}(\boldsymbol{x}),\ \ (P+Q)^{(\boldsymbol{i})}(\boldsymbol{x}) = P^{(\boldsymbol{i})}(\boldsymbol{x}) + Q^{(\boldsymbol{i})}(\boldsymbol{x}).$$

B. Multiplicity Codes

Definition 4: Let m,s be nonnegative integers, q be a prime power, and $\boldsymbol{i} = (i_1,\ldots,i_s)$ be a vector of non-negative integers. We define $\Sigma(m,s,q) = \mathbb{F}_q^{|\{\boldsymbol{i}:wt(\boldsymbol{i})<m\}|} = \mathbb{F}_q^{\binom{s+m-1}{s}}$.

Definition 5: Let m,d,s be nonnegative integers, q be a prime power, $\boldsymbol{i} = (i_1,\ldots,i_s)$ be a vector of non-negative integers, and let $\Sigma = \Sigma(m,s,q)$. For a polynomial $P(x_1,\ldots,x_s) \in \mathbb{F}_q[x_1,\ldots,x_s]$, we define the order m evaluation of P at $\boldsymbol{w} \in \mathbb{F}_q^s$, denoted by $P^{(<m)}(\boldsymbol{w})$, to be the vector

$$P^{(<m)}(\boldsymbol{w}) = \big(P^{(\boldsymbol{i})}(\boldsymbol{w})\big)_{\boldsymbol{i}:wt(\boldsymbol{i})<m} \in \Sigma.$$

*Definition 6: Let m,d,s be nonnegative integers, q be a prime power, and let $\Sigma = \Sigma(m,s,q)$. The **multiplicity code** $\mathcal{C}_M(m,d,s,q)$ of order m evaluations of degree d polynomials in s variables over $\mathbb{F}_q$ is defined as follows. The code is over the alphabet Σ, and has length q^s. The coordinates are indexed by elements of $\mathbb{F}_q^s$. For each polynomial $P(\boldsymbol{x}) \in \mathbb{F}_q[x_1,\ldots,x_s]$ with $\deg(P) \leqslant d$, there is a codeword in $\mathcal{C}_M$ given by:*

$$Enc_{m,d,s,q}(P) = \big(P^{(<m)}(\boldsymbol{w})\big)_{\boldsymbol{w}\in\mathbb{F}_q^s} \in (\Sigma)^{q^s}.$$

That is,

$$\mathcal{C}_M(m,d,s,q) = \left\{ \left(P^{(<m)}(\boldsymbol{w})\right)_{\boldsymbol{w}\in\mathbb{F}_q^s} \in \Sigma^{q^s} : \ P \in \mathbb{F}_q[\boldsymbol{x}], \deg(P) \leqslant d \right\}.$$

The following lemma, which calculates the rate and relative distance of multiplicity codes, was proved in [15].

Lemma 7 [15, Lemma 9]: The multiplicity code $\mathcal{C}_M(m,d,s,q)$ has relative minimum distance at least $\delta = 1 - \frac{d}{mq}$ and rate $\frac{\binom{d+s}{s}}{\binom{s+m-1}{s}q^s}$.

Lastly, we note that since the multiplicity code $\mathcal{C}_M(m,d,s,q)$ is a linear code it can also be a systematic code and thus for the rest of the paper we assume these codes to be systematic; for more details see [34, Lemma 2.3]. For the rest of the paper and unless stated otherwise, we assume that m, d, s, q are positive integers.

The next example demonstrates the above definitions.

Example 8: Let $q = s = 2$, $d = 3$, and $m = 4$. Thus we get that

$$\mathcal{C}_M(4,3,2,2) = \{\left(P^{(<4)}(\boldsymbol{w})\right)_{\boldsymbol{w}\in\mathbb{F}_q^2} \in \Sigma^{q^2} : P \in \mathbb{F}_2[\boldsymbol{x}], \deg(P) \leqslant 3\}.$$

If we take $P(x_1,x_2) = x_1x_2 + x_1x_2^2$, then

$$\begin{aligned} P(\boldsymbol{x}+\boldsymbol{z}) &= P(x_1+z_1, x_2+z_2) \\ &= (x_1+z_1)(x_2+z_2) + (x_1+z_1)(x_2+z_2)^2 \\ &= x_1x_2 + x_1x_2^2 + (x_2+x_2^2)z_1 + x_1z_2 + z_1z_2 \\ &\quad + x_1z_2^2 + z_1z_2^2. \end{aligned}$$

Therefore $P(\boldsymbol{x})$ is encoded to $\left(P^{(<4)}(\boldsymbol{w})\right)_{\boldsymbol{w}\in\mathbb{F}_2^2}$ where

$$\begin{aligned} &\left(P^{(<4)}(\boldsymbol{w})\right) \\ &= \big(P^{(0,0)}, P^{(1,0)}, P^{(0,1)}, P^{(2,0)}, P^{(1,1)}, P^{(0,2)}, \\ &\quad P^{(3,0)}, P^{(2,1)}, P^{(1,2)}, P^{(0,3)}\big) \\ &= \big(w_1w_2 + w_1w_2^2, w_2 + w_2^2, w_1, 0, 1, w_1, 0, 0, 1, 0\big). \end{aligned}$$

IV. PIR Codes From Multiplicity Codes

In [15], multiplicity codes were used to construct *locally decodable codes* (*LDC*) in order to retrieve the value of a single symbol from the information symbols with high probability, given that at most a fixed fraction of the symbols has errors. Since we are not concerned with errors, we modify the recovering procedure so that each information symbol has a large number of disjoint recovering sets. For this end, we establish several properties on interpolation sets of polynomials which will help us later to construct the recovering sets, and thus PIR and batch codes.

A. Interpolation Sets of Polynomials

We start with the following definition of interpolating sets.

Definition 9: For $P(\boldsymbol{x}) \in \mathbb{F}_q[x_1,\ldots,x_s]$ and $R \subseteq \mathbb{F}_q^s$, we say that R is an interpolation set of $P(\boldsymbol{x})$ if for every polynomial $Q(\boldsymbol{x})$ such that $P(\boldsymbol{x}) = Q(\boldsymbol{x})$ for every $\boldsymbol{x} \in R$, it holds that $P(\boldsymbol{x}) = Q(\boldsymbol{x})$ for every $\boldsymbol{x} \in \mathbb{F}_q^s$.

Next, we list several basic properties on the interpolation sets of polynomials. For the completeness of the results in the paper, we present these proofs in Appendix A.

Lemma 10: Let $P(\boldsymbol{x}) \in \mathbb{F}_q[x_1,\ldots,x_s]$, with $\deg(P) \leqslant d$. Let $A_1,\ldots,A_s$ be subsets of $\mathbb{F}_q$ such that $|A_i| = d+1$. Then the set $A = A_1 \times \cdots \times A_s$ is an interpolation set of $P(\boldsymbol{x})$.

According to Lemma 10, we can now generate a large number of disjoint interpolation sets for homogenous polynomials.

Lemma 11: Let $P(\boldsymbol{x}) \in \mathbb{F}_q[x_1,\ldots,x_s]$ be a homogeneous polynomial[1] *such that $\deg(P) = d$. Let $A_1,\ldots,A_{s-1}$ be subsets of $\mathbb{F}_q$ such that $|A_i| = d+1$. Then the set $A = A_1 \times \cdots \times A_{s-1} \times \{1\}$ is an interpolation set of $P(\boldsymbol{x})$, where $1 \in \mathbb{F}_q$ is the unitary element of the field.*

The following definition will be used in the construction of recovering sets for multiplicity codes.

*Definition 12: Let $\mathbb{F}_q$ be a field, and $S_1, S_2 \subseteq \mathbb{F}_q^s$ where s is a positive integer. We say that the sets S_1 and S_2 are **disjoint under multiplication** if for every $x \in S_1$ and $\alpha \in \mathbb{F}_q \setminus \{0\}$ it holds that $\alpha x \notin S_2$.*

Lemma 13: Let $P(\boldsymbol{x}) \in \mathbb{F}_q[x_1,\ldots,x_s]$ be a homogeneous polynomial such that $\deg(P) = d$. Then there exist $\lfloor \frac{q}{d+1} \rfloor^{s-1}$ interpolation sets of $P(\boldsymbol{x})$, each of size $(d+1)^{s-1}$, which are mutually disjoint under multiplication.

B. Construction of Recovering Sets

Now we are in a good position to present the recovering procedure for multiplicity codes. First, we show a general structure of the recovering sets, and then we argue that many disjoint sets can be constructed this way.

Theorem 14: Let m, d, s, q be positive integers such that $\frac{d}{m} < q-1$, and $\mathcal{C}_M(m,d,s,q)$ is the multiplicity code with these parameters of length q^s over the field $\mathbb{F}_q^{\binom{s+m-1}{s}}$. Let $A \subseteq \mathbb{F}_q^s$ be an interpolation set for homogeneous polynomials of degree at most $m-1$. Then, for every $\boldsymbol{y} = (y_{\boldsymbol{w}})_{\boldsymbol{w}\in\mathbb{F}_q^s} \in \mathcal{C}_M(m,d,s,q)$, and for every $\boldsymbol{w}_0 \in \mathbb{F}_q^s$, the set of coordinates indexed by the set

$$R = \{\boldsymbol{w}_0\} + \mathbb{F}_q A \triangleq \{\boldsymbol{w}_0 + \lambda \boldsymbol{v} : \boldsymbol{v} \in A, \lambda \in \mathbb{F}_q \setminus \{0\}\}$$

is a recovering set for the symbol $y_{\boldsymbol{w}_0}$.

Proof: The proof follows similar ideas to the one from [15]. Recall that every codeword $\boldsymbol{y} = (y_{\boldsymbol{w}})_{\boldsymbol{w}\in\mathbb{F}_q^s} \in \mathcal{C}_M(m,d,s,q)$ corresponds to a polynomial $P(\boldsymbol{x}) \in \mathbb{F}_q[\boldsymbol{x}]$, of degree at most d, where for all $\boldsymbol{w} \in \mathbb{F}_q^s$, $y_{\boldsymbol{w}} = P^{(<m)}(\boldsymbol{w})$. We now review the recovering procedure, which will eventually prove the theorem's statement. Every vector $\boldsymbol{v}$ in the interpolation set A is called a *direction* and will correspond to a line containing $\boldsymbol{w}_0$ in the direction $\boldsymbol{v}$. Reading the order m evaluations of the polynomial $P(\boldsymbol{x})$ at these lines will enable us to recover the value of $P^{(<m)}(\boldsymbol{w}_0)$. This procedure consists of two steps, described as follows.

Step 1: For every direction $\boldsymbol{v} \in A$, define the following univariate polynomial

$$p_{\boldsymbol{v}}(\lambda) = P(\boldsymbol{w}_0 + \lambda \boldsymbol{v}) \in \mathbb{F}_q[x]. \tag{1}$$

[1] We say that $P(\boldsymbol{x}) \in \mathbb{F}_q[x_1,\ldots,x_s]$ is homogeneous if all the monomials of $P(\boldsymbol{x})$ have the same total degree.

Since the values $P(\boldsymbol{w}_0 + \lambda \boldsymbol{v})$ for all $\lambda \in \mathbb{F}_q \setminus \{0\}$ are known, and $\deg(p_{\boldsymbol{v}}) \leqslant d$, one can prove, as in [15], that $p_{\boldsymbol{v}}(\lambda)$ is unique, and thus can be recovered.

For completeness, we prove that $p_{\boldsymbol{v}}(\lambda)$ is indeed unique. Assume in the contrary that there exist two different polynomials $p_1(\lambda)$ and $p_2(\lambda)$, of degree at most d, such that

$$p_1(\lambda) = p_2(\lambda) = P(\boldsymbol{w}_0 + \lambda \boldsymbol{v}) \quad \text{for all } \lambda \in \mathbb{F}_q \setminus \{0\}.$$

According to the definition of Hasse derivative, we have that

$$P(\boldsymbol{w}_0 + (\lambda_0 + \lambda)\boldsymbol{v}) = p_r(\lambda_0 + \lambda) = \sum_{j=0}^{d} p_r^{(j)}(\lambda_0)(\lambda)^j,$$

and

$$P(\boldsymbol{w}_0 + (\lambda_0 + \lambda)\boldsymbol{v}) = \sum_{\boldsymbol{i}:wt(\boldsymbol{i}) \leqslant d} P^{(\boldsymbol{i})}(\boldsymbol{w}_0 + \lambda_0 \boldsymbol{v})(\lambda \boldsymbol{v})^{\boldsymbol{i}},$$

for $r \in \{1, 2\}$. Therefore, for $0 \leqslant j \leqslant d$,

$$p_r^{(j)}(\lambda_0) = \sum_{\boldsymbol{i}:wt(\boldsymbol{i})=j} P^{(\boldsymbol{i})}(\boldsymbol{w}_0 + \lambda_0 \boldsymbol{v})\boldsymbol{v}^{\boldsymbol{i}}.$$

Thus, given the values of $P^{(<m)}(\boldsymbol{w}_0 + \lambda \boldsymbol{v})$ we can calculate the values of

$$p_r^{(<m)}(\lambda) = (p_r^{(0)}(\lambda), p_r^{(1)}(\lambda), \ldots, p_r^{(m-1)}(\lambda)).$$

It follows that $p_1^{(<m)}(\lambda) = p_2^{(<m)}(\lambda)$ for every $\lambda \in \mathbb{F}_q \setminus \{0\}$. Those two polynomials can be thought of as codewords of $\mathcal{C}_M(m, d, s = 1, q)$. From Lemma 7, we know that the distance of the code $\mathcal{C}_M(m, d, s = 1, q)$ is $q(1 - \frac{d}{mq}) = q - \frac{d}{m} > 1$. Therefore, the two polynomials p_1, p_2 are identical. Thus, one can uniquely recover the polynomial, $p_{\boldsymbol{v}}(\lambda)$, which we now denote by

$$p_{\boldsymbol{v}}(\lambda) = \sum_{j=0}^{d} c_{\boldsymbol{v},j} \lambda^i.$$

Step 2: From (1), one can get that

$$p_{\boldsymbol{v}}(\lambda) = \sum_{\boldsymbol{i}} P^{(\boldsymbol{i})}(\boldsymbol{w}_0)\boldsymbol{v}^{\boldsymbol{i}} \lambda^{wt(\boldsymbol{i})} = \sum_{j=0}^{d} c_{\boldsymbol{v},j} \lambda^j,$$

and therefore we get for all $0 \leqslant j \leqslant d$,

$$\sum_{\boldsymbol{i}:wt(\boldsymbol{i})=j} P^{(\boldsymbol{i})}(\boldsymbol{w}_0)\boldsymbol{v}^{\boldsymbol{i}} = c_{\boldsymbol{v},j}.$$

Considering only the first m of these equations, we get that $u_{\boldsymbol{i}} = P^{(\boldsymbol{i})}(\boldsymbol{w}_0)$ is a solution for the following set of equations

$$\sum_{\boldsymbol{i}:wt(\boldsymbol{i})=j} u_{\boldsymbol{i}} \boldsymbol{v}^{\boldsymbol{i}} = c_{\boldsymbol{v},j}, \quad 0 \leqslant j < m \leqslant d, \ \boldsymbol{v} \in A. \tag{2}$$

Now we prove that the equations system (2) has a unique solution, and therefore the set R is a recovery set. Indeed, if we denote $Q_j(\boldsymbol{x}) = \sum_{\boldsymbol{i}:wt(\boldsymbol{i})=j} u_{\boldsymbol{i}} \boldsymbol{x}^{\boldsymbol{i}} \in \mathbb{F}_q[x_1, \ldots, x_s]$ where $0 \leqslant j < m$, we get that the equations in (2) are equivalent to

$$Q_j(\boldsymbol{v}) = c_{\boldsymbol{v},j}$$

for every $\boldsymbol{v} \in A$. But since for every j we know that Q_j is a homogeneous polynomial of degree j, and A is an interpolation set for homogeneous polynomials of degree at most $m - 1$, we get that the polynomial $Q_j(\boldsymbol{x})$ is unique. Therefore, we can recover the value of $P^{(<m)}(\boldsymbol{w}_0)$ by solving the equations system (2). ∎

Example 15: Let $q = 4$, $m = s = 2$, $d = 3$. First, we denote $\mathbb{F} = \mathbb{F}_4 = \{0, 1, \alpha, \beta = 1 + \alpha\}$. Let us take the codeword corresponding to the polynomial $P(x_1, x_2) = x_1x_2 + x_1x_2^2$ inside the code $\mathcal{C}_M(2, 3, 2, 4)$. According to Example 8,

$$P^{(<2)}(x_1, x_2) = (x_1x_2 + x_1x_2^2, x_2 + x_2^2, x_1).$$

Therefore, the polynomial $P(x_1, x_2)$ is encoded to the codeword $(w_1w_2 + w_1w_2^2, w_2 + w_2^2, w_1)_{\boldsymbol{w} \in \mathbb{F}_4^2}$. Now, assume we want to recover the codeword entry at the point $\boldsymbol{w} = (1, 0) \in \mathbb{F}_4^2$, i.e., we want to recover the value of

$$P^{(<2)}(1, 0) = \left(P^{(0,0)}(1, 0), P^{(1,0)}(1, 0), P^{(0,1)}(1, 0)\right).$$

First, we consider the line at direction $\boldsymbol{v} = (1, 0)$,

$$R = \{(1, 0) + \lambda(1, 0) : \lambda \in \mathbb{F} \setminus \{0\}\} = \{(0, 0), (\alpha, 0), (\beta, 0)\}.$$

We define $p(\lambda) = P(\boldsymbol{w} + \lambda \boldsymbol{v})$. From the given codeword, we get that:

$$P^{(<2)}(0, 0) = (0, 0, 0),$$
$$P^{(<2)}(\alpha, 0) = (0, 0, \alpha),$$
$$P^{(<2)}(\beta, 0) = (0, 0, \beta).$$

Therefore, $p(1) = p(\alpha) = p(\beta) = 0$, and the unique polynomial of degree at most 3 which satisfies this condition is the zero polynomial, therefore $p(\lambda) \equiv 0$, and we can already recover $P^{(0,0)}(1, 0) = p(0) = 0$. Now, in order to recover $P^{(1,0)}(1, 0)$, we notice that

$$p(\lambda) = P(\boldsymbol{w} + \lambda \boldsymbol{v}) = \sum_{\boldsymbol{i}} P^{(\boldsymbol{i})}(\boldsymbol{w})\boldsymbol{v}^{\boldsymbol{i}} \lambda^{wt(\boldsymbol{i})}.$$

Since $p(\lambda) \equiv 0$, we get that

$$P^{(1,0)}(1, 0)v^{(1,0)} + P^{(0,1)}(1, 0)v^{(0,1)} = 0,$$

therefore $P^{(1,0)}(1, 0) = 0$ since $v^{(0,1)} = 0$. In order to recover $P^{(0,1)}(1, 0)$, we consider a new line in the direction $\boldsymbol{v}' = (0, 1)$,

$$R' = \{(1, 0) + \lambda(0, 1) : \lambda \in \mathbb{F} \setminus \{0\}\} = \{(1, 0), (1, \alpha), (1, \beta)\}.$$

Repeating the same steps as above for the new direction $\boldsymbol{v}'$, we can get that $P^{(0,1)}(1, 0) = 1$. Thus, we have recovered that $P^{(<2)}(1, 0) = (0, 0, 1)$.

The next theorem calculates the number of disjoint recovering sets that can be constructed for every coordinate.

Theorem 16: Let $\boldsymbol{y} = (y_{\boldsymbol{w}})_{\boldsymbol{w} \in \mathbb{F}_q^s} \in \mathcal{C}_M(m, d, s, q)$ *and* $\boldsymbol{w}_0 \in \mathbb{F}_q^s$. *Then,* $y_{\boldsymbol{w}_0}$ *has* $\lfloor \frac{q}{m} \rfloor^{s-1}$ *mutually disjoint recovering sets.*

Proof: According to Theorem 14, every interpolation set A for homogeneous polynomials of degree $m - 1$ defines a recovering set, which consists of the lines containing $\boldsymbol{w}_0$ in the directions of $\boldsymbol{v}$ for all $\boldsymbol{v} \in A$. Therefore, in order to get disjoint recovering sets, all we need to do is to pick different lines. According to Lemma 13, there are $\lfloor \frac{q}{m} \rfloor^{s-1}$ interpolation

sets for homogeneous polynomials of degree $m-1$ which are mutually disjoint under multiplication. This means that each line cannot appear in two sets, thus the recovering sets defined by these interpolation sets are disjoint. ∎

Now we can conclude with the following corollary, which follows directly from Theorem 16.

Corollary 17: For all m, d, s, q such that $\frac{d}{m} < q-1$, the code $\mathcal{C}_M(m,d,s,q)$ is a k-PIR code $[q^s, n, k]_Q^P$, where $n = \frac{\binom{d+s}{s}}{\binom{s+m-1}{s}}$, $k = \lfloor \frac{q}{m} \rfloor^{s-1}$, and $Q = q^{\binom{s+m-1}{s}}$.

The next theorem summarizes the results in this section.

Theorem 18: For every positive integer $s \geqslant 2$, $0 < \alpha < 1$, and n sufficiently large, there exists a k-PIR code $[N, n, k]_Q^P$, over the field $\mathbb{F}_Q$, of dimension n such that the redundancy, $r = N - n$, the availability parameter, k, and the field size Q satisfy

$$\begin{aligned} k &= \Theta(n^{(1-\frac{1}{s})(1-\alpha)}), \\ Q &= n^{\Theta(n^{\alpha})}, \\ r &= \mathcal{O}(n^{1-\frac{\alpha}{s}}). \end{aligned}$$

In particular, for all $0 \leqslant \epsilon < 1$, integer $s > \frac{1}{1-\epsilon}$ and n sufficiently large, there exists a k-PIR code $[N, n, k]_Q^P$, where $k = n^{\epsilon}$ and the redundancy satisfies $N - n = \mathcal{O}(\delta_s(\epsilon))$, where $\delta_s(\epsilon) = \mathcal{O}(1 - \frac{1}{s} + \frac{\epsilon}{s-1})$. Moreover, for all $0 \leqslant \epsilon < 1$,

$$r_P\big(n, k = \Theta(n^{\epsilon})\big)_Q = \mathcal{O}\big(n^{\delta(\epsilon)}\big),$$

where $\delta(\epsilon) = 1 - \frac{1}{s^*} + \frac{\epsilon}{s^*-1}$, and $s^* = \lfloor \frac{2}{1-\epsilon} \rfloor$.

Proof: Let q be a prime power such that, $m = \lfloor q^{\alpha} \rfloor$ and $d = m(q-2)$. We choose n such that according to Corollary 17, the code $\mathcal{C}_M(m,d,s,q)$ is a k-PIR code $[N = q^s, n, k]_Q^P$ where n, k, Q satisfy

$$\begin{aligned} n &= \frac{\binom{d+s}{s}}{\binom{s+m-1}{s}} = \Theta(N), \\ k &= \Theta(q^{(s-1)(1-\alpha)}) = \Theta(N^{(1-\frac{1}{s})(1-\alpha)}) = \Theta(n^{(1-\frac{1}{s})(1-\alpha)}), \\ Q &= q^{\binom{s+m-1}{s}} = q^{\Theta(q^{\alpha s})} = q^{\Theta(n^{\alpha})} = n^{\Theta(n^{\alpha})}. \end{aligned}$$

It remains to prove the claim about the redundancy. According to Lemma 7, the redundancy of the code satisfies

$$\begin{aligned} r &= q^s - \frac{\binom{d+s}{s}}{\binom{s+m-1}{s}} \leqslant q^s - \frac{d^s}{(m+s)^s} \\ &= \frac{(m+s)^s q^s - m^s (q-2)^s}{(m+s)^s} \\ &\leqslant \frac{(m+s)^s q^s - m^s (q-2)^s}{m^s} \\ &= \mathcal{O}(\frac{q^s}{m}) = \mathcal{O}(q^{s-\alpha}) \\ &= \mathcal{O}(N^{1-\frac{\alpha}{s}}) \\ &= \mathcal{O}(n^{1-\frac{\alpha}{s}}). \end{aligned}$$

Now we prove the second part of the claim. We consider a fixed s. According to the first part of the proof, for every $0 < \alpha < 1$, there exists a k-PIR code of dimension n and redundancy r such that $k = \Theta(n^{(1-\frac{1}{s})(1-\alpha)})$ and $r = \mathcal{O}(n^{1-\frac{\alpha}{s}})$.

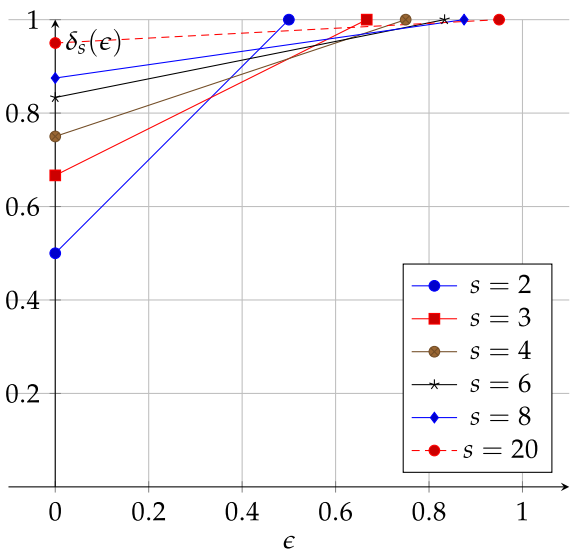

Fig. 1. Asymptotic results for non-binary PIR codes.

For $\epsilon < 1 - 1/s$, we denote $\epsilon = (1-\frac{1}{s})(1-\alpha)$, or $\alpha = 1 - \frac{\epsilon s}{s-1}$, and we get

$$r = \mathcal{O}(n^{1-\frac{\alpha}{s}}) = \mathcal{O}(n^{1-\frac{1}{s}+\frac{\epsilon}{s-1}}) = \mathcal{O}(n^{\delta_s(\epsilon)}).$$

Thus, we deduce that $\delta(\epsilon) = \min_{s: s > \frac{1}{1-\epsilon}} \{\delta_s(\epsilon)\}$. Since $\delta_s(\epsilon)$ are lines for every s, one can prove that for a given value of ϵ, the value of s^* that minimizes the value of δ satisfies

$$\frac{s^*-2}{s^*} \leqslant \epsilon < \frac{s^*-1}{s^*+1},$$

that is $s^* = \lfloor \frac{2}{1-\epsilon} \rfloor$.

Lastly, in case n is not of the form $\binom{d+s}{s} / \binom{s+m-1}{s}$ we can choose the parameters q, m, d such that $\binom{d+s}{s} / \binom{s+m-1}{s} > n$. Then, we apply the construction while appending the information symbols with the missing number of zero symbols but without storing them since their values are known in advance to be zero. ∎

In Fig. 1 we plot the curves of $\delta_s(\epsilon)$ (which are defined in Theorem 18) for various values of s for non-binary PIR codes.

C. PIR Codes Over the Binary Field

Now we use our last result in order to construct binary k-PIR codes. The main idea is to convert every symbol of the field $\mathbb{F}_Q$ to $\log(Q)$ binary symbols. We say that $f(n) = \Omega(n^{a^-})$ if for all $\tau > 0$, $f(n) = \Omega(n^{a-\tau})$. Similarly we define $f(n) = \mathcal{O}(n^{a^+})$ if for all $\tau > 0$, $f(n) = \mathcal{O}(n^{a+\tau})$. The next theorem is a direct result from Theorem 18 and for completeness, its proof appears in Appendix B.

Theorem 19: For every positive integer $2 \leqslant s$, $0 < \alpha < 1$, and n sufficiently large, there exists a binary k-PIR code $[N, n, k]_2^P$, of dimension n such that the redundancy, $r = N - n$, and the availability parameter, k, satisfy

$$\begin{aligned} k &= \Theta\big((n/\log(n))^{(1-\frac{1}{s})\frac{1-\alpha}{1+\alpha}}\big), \\ r &= \mathcal{O}\big(n^{1-\frac{\alpha}{s(1+\alpha)}} (\log(n))^{\frac{\alpha}{s(1+\alpha)}}\big). \end{aligned}$$

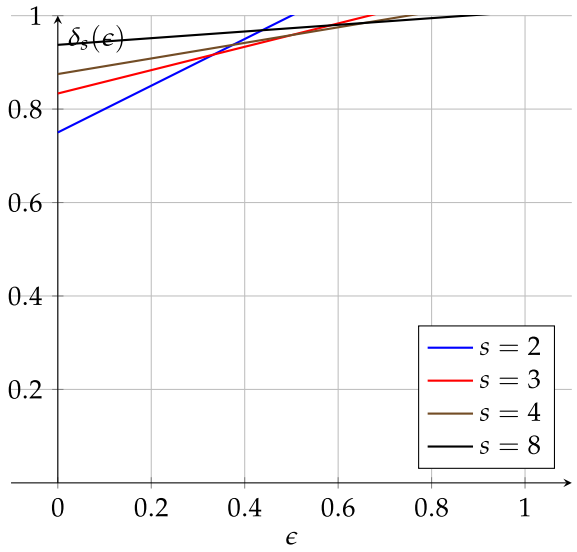

Fig. 2. Asymptotic results for binary PIR codes.

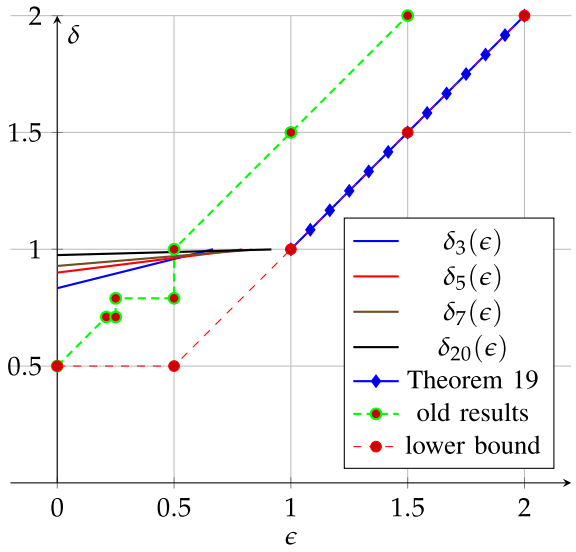

Fig. 3. Asymptotic results for binary PIR codes.

In particular, for $0 \leqslant \epsilon < 1$*, it holds that*

$$r_P\left(n, k = \Omega(n^{\epsilon^-})\right) = \mathcal{O}\left(n^{\delta(\epsilon)^+}\right),$$

where

$$\delta(\epsilon) = \min_{s: s > \frac{1}{1-\epsilon}} \{\delta_s(\epsilon)\},$$
$$\delta_s(\epsilon) = 1 - \frac{s(1-\epsilon)-1}{2s(s-1)}.$$

Fig. 2 shows the curves of $\delta_s(\epsilon)$ (which are defined in Theorem 19) for various values of s, for binary PIR codes. The following corollary follows from the last theorem.

Corollary 20: For all $0 \leqslant \epsilon < 1$ *and* n *sufficiently large, there exists a binary* k*-PIR code* $[N, n, k]_2^P$*, of dimension* n *such that* $k = \Theta(n^\epsilon)$*, and the redundancy is* $r = N - n = o(n)$ *(i.e., the rate converges to 1).*

Proof: The claim follows immediately from the last theorem by noticing that $r = o(n)$ since $\delta_s(\epsilon) < 1$. ∎

The analysis so far dealt with constructing k-PIR when $k = \Theta(n^\epsilon)$ and $0 \leqslant \epsilon < 1$. Now we show how to use these results to construct k-PIR codes for $\epsilon \geqslant 1$. The idea is to concatenate a sufficient copies of k'-PIR codes, when $k' = O(n^{1^-})$ such that each bit will have k recovering sets. The following lemma was proved in [11].

Lemma 21 [11, Lemma 12]: Let n k*, and* k' *be positive integers. Then,* $P(n, k+k') \leqslant P(n,k) + P(n,k')$.

If follows that for any positive integers n, k, and k' such that $k' < k$, $P(n,k) \leqslant \lceil \frac{k}{k'} \rceil P(n,k')$. Thus we get the following theorem.

Theorem 22: For all $\epsilon \geqslant 1$ *and* n *sufficiently large, there exists a binary* k*-PIR code* $[N, n, k]^P$*, of dimension* n *such that* $k = \Theta(n^\epsilon)$ *and* $N = \mathcal{O}(n^{\epsilon^+})$.

The length achieved by the PIR construction in Theorem 22 is nearly optimal. Recall that the redundancy of k-PIR codes is $\Omega(k)$ since every non-trivial recovering set must contain at least one redundancy bit.

Fig. 3 summarizes the results of binary PIR codes we achieved in this section together with the previous results. We denote $r_P\left(n, k = n^\epsilon\right) = \mathcal{O}\left(n^{\delta(\epsilon)}\right)$ and plot upper and lower bounds for $\delta(\epsilon)$. We plot the curves $\delta_s(\epsilon)$ for $s = 3, 5, 7, 20$ from Theorem 19 as well as the results for $\epsilon \geqslant 1$ from Theorem 22, which provide an upper bound for $\delta(\epsilon)$. Since Reed-Muller codes have rate at most $1/2$, their upper bounds were not added to the figure, as they will always result with $\delta = 1$ for any $\epsilon < 1$. The lower bound on the redundancy is given by $\Omega(\max\{k, \sqrt{n}\})$. Finally, we note that according to this lower bound and Theorem 22, we get that $\epsilon \leqslant \delta(\epsilon) \leqslant \epsilon + \rho$ when $1 \leqslant \epsilon$ for any $\rho > 0$.

V. Batch Codes From Multiplicity Codes

It turns out that multiplicity codes can be also an excellent tool to construct batch codes. Unlike the PIR case, recovering different entries in the codeword will cause intersection in the corresponding lines, and thus intersecting recovering sets. In order to overcome this obstacle, we reduce the degree d of the polynomials such that a fewer number of points is needed from every line. This will allow different lines to avoid points which are used by other lines. That way, every recovering set can "drop out" points which are used by other sets, resulting in disjoint recovering sets.

Lemma 23: For all m, s, q, d, k *such that* $d \leqslant m(q - km^{s-1} - 2)$ *and* $k \leqslant \lfloor \frac{q}{m} \rfloor^{s-1}$*, the code* $\mathcal{C}_M(m, d, s, q)$ *is a* k*-batch code* $[q^s, n, k]_Q^B$*, where* $n = \frac{\binom{d+s}{s}}{\binom{s+m-1}{s}}$ *and* $Q = q^{\binom{s+m-1}{s}}$.

Proof: The claim regarding the code dimension and field size can be proven in a similar way to the PIR case. Now we prove that every request of size k can be recovered. To this end, we show that each recovery set can drop out the points which are used by other sets. As we saw in the recovering procedure for PIR codes, every recovering set contains m^{s-1} different lines. Since different lines can intersect on at most one point, and there are k recovering sets (we can assume that every line

appears in at most one set), we get that every recovering set has km^{s-1} points which are used by other sets. Therefore, it suffices to prove that step 1 in the recovering procedure can be completed even when km^{s-1} points on the line are not used. However, according to Lemma 7, the minimum distance of $\mathcal{C}_M(m, d, s = 1, q)$ is $d = q - \frac{d}{m} > km^{s-1} + 1$, and therefore it can be shown in a very similar way to PIR codes, that the polynomial $p_{\boldsymbol{v}}$ in step 1 can be uniquely recovered, and thus also step 2 can be completed. ∎

Unlike the PIR case, it turns out that only the value $s = 2$ is useful for batch codes. The reason is that large values of the parameter s cause each recovering set to contain more lines, and thus the number of intersecting points between different sets will also increase. Therefore, as we reduce the degree d of the polynomials to be able to ignore these points, the resulting degree becomes substantially smaller, which results in codes with low rate. The following theorem uses Lemma 23 with $s = 2$ to construct batch codes from multiplicity codes. We present the proofs of Theorem 24 and Theorem 25 in Appendix C.

Theorem 24: For every $0 < \alpha < 0.5$ and n sufficiently large, there exists a k-batch code $[N, n, k]_Q^B$ over $\mathbb{F}_Q$ of dimension n such that the redundancy, $r = N - n$, the availability parameter, k, and the field size Q satisfy

$$\begin{aligned} k &= \Theta(n^{0.5-\alpha}), \\ r &= \mathcal{O}(n^{1-\frac{\alpha}{2}}), \\ Q &= n^{\Theta(n^{\alpha})}. \end{aligned}$$

In particular, for $0 < \epsilon < 0.5$, it holds that

$$r_B\big(n, k = \Theta(n^{\epsilon})\big)_Q = \mathcal{O}\big(n^{\delta(\epsilon)}\big),$$

where $\delta(\epsilon) = \frac{3}{4} + \frac{\epsilon}{2}$.

As in the PIR case, the last result can be extended for binary batch codes.

Theorem 25: For every $0 < \alpha < 0.5$ and n sufficiently large, there exists a binary k-batch code $[N, n, k]^B$ of dimension n such that the redundancy, $r = N - n$, and the availability parameter, k, satisfy

$$\begin{aligned} k &= \Theta\big((n/\log(n))^{0.5-\alpha}\big), \\ r &= \mathcal{O}\big(n^{1-\frac{\alpha}{3}}(\log(n))^{\frac{\alpha}{3}}\big). \end{aligned}$$

In particular, for $0 < \epsilon < 0.5$, it holds that

$$r_B\big(n, k = \Theta(n^{\epsilon^-})\big) = \mathcal{O}\big(n^{\delta(\epsilon)^+}\big),$$

where $\delta(\epsilon) = \frac{5}{6} + \frac{\epsilon}{3}$, and $r_B\big(n, k = \Theta(n^{\epsilon})\big) = o(n)$.

The following corollary follows from the last theorem.

Corollary 26: For all $0 \leqslant \epsilon < 0.5$ and n sufficiently large, there exists a binary k-batch code $[N, n, k]^B$, of dimension n such that $k = \Theta(n^{\epsilon})$ and the redundancy $r = N - n = o(n)$ (i.e., the rate converges to 1).

As in the PIR case, one can extend these batch codes to the case where $\epsilon \geqslant 0.5$ and construct binary batch codes $[N, n, k]^B$ such that $k = \Theta(n^{\epsilon})$ and $N = \mathcal{O}(n^{0.5+\epsilon^+})$, for $\epsilon \geqslant 0.5$. However, in the next section we show another method to construct batch codes which achieve better results in the case $\epsilon \geqslant 0.5$. In particular, we show that $r_B(n, k = n^{\epsilon}) = o(n)$ when $0 < \epsilon < 1$, and $r_B(n, k = n^{\epsilon}) = \mathcal{O}(n^{\epsilon^+})$ when $1 \leqslant \epsilon$.

VI. Construction of Batch Codes From PIR Codes

In this section we present a method that shows how to construct a batch code from a given PIR code. Then, using the construction of PIR codes in section IV, we show a construction of $k = n^{\epsilon}$-batch codes of dimension n and rate approaching 1 where $\epsilon < 1$, i.e., $r_B(n, n^{\epsilon}) = o(n)$ for $\epsilon < 1$. We present the results for the binary field, however the extension to the non-binary case is trivial.

The techniques in this section use the notion of batch codes (PIR codes) with restricted size for the recovering sets [35]. Formally, a k-PIR code, k-batch code, in which the size of each recovering set is at most r will be called an (r, k)-*PIR code*, (r, k)-*batch code*, respectively. First, we show how to construct PIR codes with restricted size of recovery sets from general PIR codes. Then we show how to construct batch codes from PIR codes with restricted size of recovery sets.

Lemma 27: Let $\mathcal{C}$ be an $[N, n, k]^P$ PIR code. Then, $\mathcal{C}$ is also an $(r = \frac{2N}{k}, k' = \frac{k}{2})$-PIR code of dimension n and length N.

Proof: We prove that for every bit there exist $k' = k/2$ disjoint recovery sets of size at most $r = \frac{2N}{k}$. Since $\mathcal{C}$ is an $[N, n, k]^P$ code, we get that every bit has k disjoint recovery sets. Assume by contradiction that there exist $k/2$ recovery sets of size more than r, thus since these sets are disjoint, we get that there exist at least $rk/2 + 1 = N + 1$ different bits inside these sets, which is a contradiction. ∎

The following lemma shows how to construct batch codes from PIR codes with restricted size of recovery set.

Lemma 28: Let $\mathcal{C}$ be an (r, k)-PIR code of dimension n and length N. Then, $\mathcal{C}$ is an $[N, n, k']^B$ batch code, where $k' = \lfloor k/r \rfloor$.

Proof: Let $R = \{i_1, i_2, \ldots, i_{k'}\}$ be the multiset of requested bits. Since $\mathcal{C}$ is an (r, k)-PIR code, we get that for every $j \in [k]$ there exist k disjoint recovery sets $S_1^{i_j}, S_2^{i_j}, \ldots, S_k^{i_j}$ for the bit i_j, each of size at most r. Now we show how to construct k' disjoint recovery sets for them. For i_1, we take the sets $S_1^{i_1}$. Now, assume we have chosen recovery sets for the first $\ell - 1 < k' - 1$ bits, and we show how to construct a recovery set for i_ℓ. The number of bits contained in the chosen sets is at most $(\ell - 1)r < k$, therefore there exists a recovery set of i_ℓ which does not intersect with the chosen sets. We choose this set to be the recovery set of i_ℓ. ∎

From the last two lemmas we conclude the following corollary.

Corollary 29: Let $\mathcal{C}$ be an $[N, n, k]^P$ PIR code. Then, $\mathcal{C}$ is an $[N, n, k']^B$ batch code, where $k' = \lfloor k^2/4N \rfloor$.

The next theorem establishes the result on the redundancy of batch codes.

Theorem 30: For $0 < \epsilon < 1$, it holds that

$$r_B(n, n^{\epsilon}) \leqslant r_P(n, n^{\frac{\epsilon+1}{2}}).$$

Proof: Denote $\epsilon' = \frac{\epsilon+1}{2}$. Since $\epsilon' < 1$, we get that $r_P(n, n^{\epsilon'}) = o(n)$, which means that there exist a binary PIR code $[N, n, k = n^{\epsilon'}]^P$ such that $N \leqslant 2n$. Thus, by applying Corollary 29 to that code we conclude that this code is also a batch code $[N, n, k']^B$ where $k' = \Theta(n^{\epsilon})$. ∎

The following corollary follows immediately by using the previous theorem, along with Theorem 19 from section IV.

Corollary 31: For $0 < \epsilon < 1$, it holds that

$$r_B\big(n, k = \Omega(n^{\epsilon^-})\big) = \mathcal{O}\big(n^{\delta(\frac{\epsilon+1}{2})^+}\big),$$

where

$$\delta(\epsilon) = \min_{s:s>\frac{1}{1-\epsilon}} \{\delta_s(\epsilon)\},$$
$$\delta_s(\epsilon) = 1 - \frac{s(1-\epsilon)-1}{2s(s-1)}.$$

As in the case of PIR codes, we can now use the previous result to construct batch codes when $\epsilon \geqslant 1$. Then, the following theorem follows immediately.

Theorem 32: For all $\epsilon \geqslant 1$ and n sufficiently large, there exists a binary k-batch code $[N, n, k]^B$, of dimension n such that $k = \Theta(n^{\epsilon})$ and $N = \mathcal{O}(n^{\epsilon^+})$.

VII. The Array Construction for PIR Codes

In this section we present a construction which constitutes a family of PIR codes. Then, in the next section, we show how to use this construction in order to construct good batch codes. By a slight abuse of notation, in this section we let the set $[n]$ denote the set of integers $\{0, 1, \ldots, n-1\}$.

Our point of departure is the subcube construction from [13] which was also used in [11] to construct PIR codes. The idea of this construction is to position the information bits in a two-dimensional array, and add a simple parity bit for each row and each column. More specifically, given an input $\boldsymbol{x}$ of length $n = s^2$, it is represented as a two-dimensional array of size $s \times s$ as follows

x_0	x_1	$\ldots$	x_{s-1}
x_s	x_{s+1}	$\ldots$	x_{2s-1}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_{s^2-s}	x_{s^2-s+1}	$\ldots$	x_{s^2-1}

This array is encoded to the following $(s+1) \times (s+1)$ array

x_0	x_1	$\ldots$	x_{s-1}	r_0
x_s	x_{s+1}	$\ldots$	x_{2s-1}	r_1
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_{s^2-s}	x_{s^2-s+1}	$\ldots$	x_{s^2-1}	r_{s-1}
c_0	c_1	$\ldots$	c_{s-1}	

where r_i, c_i is the sum of the bits in the ith row, column, respectively. It is easy to see that every bit has 3 recovering sets; one is the bit itself, and another two recovering sets from the row and column that contain that bit.

In this work we extend this construction by following the approach which was described in [4] to correct multiple phased burst errors and erasures. Similarly, our approach here is to extend this construction by considering also the diagonals of the array. As there are approximately s diagonals, this will greatly increase the number of recovering sets. However, we will have to guarantee that using the diagonals will still result with disjoint recovering sets.

Before presenting the construction, which we will refer as the *Array Construction*, we formally define the diagonal sets that will be used in the array. We use the notation $\langle x \rangle_m$ to denote the value of $(x \bmod m)$.

Definition 33: Let A be an $r \times p$ array, with indices $(i, j) \in [r] \times [p]$. For $s \in [p]$ we define the following set of sets:

$$P_s(r, p) = \{D_{s,0}, D_{s,1}, \ldots, D_{s,p-1}\},$$

where for $t \in [p]$,

$$D_{s,t} = \{(0, t), (1, \langle t+s \rangle_p), \ldots, (r-1, \langle t+(r-1)s \rangle_p)\}. \tag{3}$$

The idea behind Definition 33 is to fix a slope $s \in [p]$ and then define p diagonal sets which are determined by the starting point on the first row and the slope. We use these sets in order to construct array codes, where every diagonal determines a parity bit for the bits on this diagonal.

Construction 34 (Array Construction): Let r, p, n, k be positive integers such that $k \leqslant p$, $n = rp$, and $S \subseteq [p]$ a subset of size k. We define the encoder $E_{r,p,S}$, as a mapping $E_{r,p,S} : \{0,1\}^n \to \{0,1\}^{k \cdot p}$ as follows. We denote $S = \{s_0, s_1, \ldots, s_{k-1}\}$ where $0 \leqslant s_0 < s_1 < \cdots < s_{k-1} \leqslant p-1$. The input vector $\boldsymbol{x} \in \{0,1\}^n$ is represented as an $r \times p$ array, that is $\boldsymbol{x} = (x_{i,j})_{(i,j)\in[r]\times[p]}$ and is encoded to the following kp redundancy bits $\rho_{\ell,t}$, for $\ell \in [k]$, and $t \in [p]$,

$$\rho_{\ell,t} = \sum_{(i,j)\in D_{s_\ell,t}} x_{i,j}.$$

We denote

$$E_{r,p,S}(\boldsymbol{x}) = (\rho_{0,0}, \ldots, \rho_{0,p-1}, \ldots, \rho_{k-1,0}, \ldots, \rho_{k-1,p-1}).$$

Lastly, the code $\mathcal{C}_A(r, p, S)$ is defined to be

$$\mathcal{C}_A(r, p, S) = \{(\boldsymbol{x}, E_{r,p,S}(\boldsymbol{x})) : \ \boldsymbol{x} \in \{0,1\}^n\}.$$

The following example demonstrates the Array Construction.

Example 35: Let $r = 3$, $p = 5$, $n = 15$, and assume we want $k = 4$ recovering sets. First, we arrange the input, $\boldsymbol{x}$, in a 3×5 table

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$	$x_{0,4}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$

We consider diagonals corresponding for slops in the set $S = \{0, 1, 2\}$. The diagonals corresponding to $s = 0$ are the columns, and so we get the following set of five sets:

$$P_0 = \{\{x_{0,0}, x_{1,0}, x_{2,0}\}, \{x_{0,1}, x_{1,1}, x_{2,1}\}, \{x_{0,2}, x_{1,2}, x_{2,2}\}, \{x_{0,3}, x_{1,3}, x_{2,3}\}, \{x_{0,4}, x_{1,4}, x_{2,4}\}\}.$$

Accordingly, we get the following parity bits

$$\rho_{0,0} = x_{0,0} + x_{1,0} + x_{2,0} \quad \rho_{0,1} = x_{0,1} + x_{1,1} + x_{2,1}$$
$$\rho_{0,2} = x_{0,2} + x_{1,2} + x_{2,2} \quad \rho_{0,3} = x_{0,3} + x_{1,3} + x_{2,3}$$
$$\rho_{0,4} = x_{0,4} + x_{1,4} + x_{2,4}$$

For $s = 1$, the following table shows the corresponding diagonals.

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$	$x_{0,4}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$

Thus we get the following set of five sets:

$$P_1 = \{\{x_{0,0}, x_{1,1}, x_{2,2}\}, \{x_{0,1}, x_{1,2}, x_{2,3}\}, \{x_{0,2}, x_{1,3}, x_{2,4}\}, \{x_{0,3}, x_{1,4}, x_{2,0}\}, \{x_{0,4}, x_{1,0}, x_{2,1}\}\},$$

and the corresponding parity bits are

$$\rho_{1,0} = x_{0,0} + x_{1,1} + x_{2,2} \quad \rho_{1,1} = x_{0,1} + x_{1,2} + x_{2,3}$$
$$\rho_{1,2} = x_{0,2} + x_{1,3} + x_{2,4} \quad \rho_{1,3} = x_{0,3} + x_{1,4} + x_{2,0}$$
$$\rho_{1,4} = x_{0,4} + x_{1,0} + x_{2,1}$$

Lastly, for $s = 2$, the following table shows the corresponding diagonals.

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$	$x_{0,4}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$

Their parity bits are the following

$$\rho_{2,0} = x_{0,0} + x_{1,2} + x_{2,4} \quad \rho_{2,1} = x_{0,1} + x_{1,3} + x_{2,0}$$
$$\rho_{2,2} = x_{0,2} + x_{1,4} + x_{2,1} \quad \rho_{2,3} = x_{0,3} + x_{1,0} + x_{2,2}$$
$$\rho_{2,4} = x_{0,4} + x_{1,1} + x_{2,3}$$

Therefore, $\boldsymbol{x}$ is encoded to

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$	$x_{0,4}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$
$\rho_{0,0}$	$\rho_{0,1}$	$\rho_{0,2}$	$\rho_{0,3}$	$\rho_{0,4}$
$\rho_{1,0}$	$\rho_{1,1}$	$\rho_{1,2}$	$\rho_{1,3}$	$\rho_{1,4}$
$\rho_{2,0}$	$\rho_{2,1}$	$\rho_{2,2}$	$\rho_{2,3}$	$\rho_{2,4}$

One can notice that every bit has 4 disjoint recovering sets. For example, the bit $x_{0,0}$ can be recovered by

$$\{x_{0,0}\}, \{\rho_{0,0}, x_{1,0}, x_{2,0}\}, \{\rho_{1,0}, x_{1,1}, x_{2,2}\}, \{\rho_{2,0}, x_{1,2}, x_{2,4}\}.$$

We first present two useful properties, whose proofs can be found in Appendix D.

Lemma 36: For all r, p, and $s \in [p]$ the set $P_s(r, p)$ is a partition of $[r] \times [p]$.

The following lemma shows that sets from different partitions can intersect on at most one element.

Lemma 37: For all $r \leqslant p$ and $S \subseteq [p]$, if one of the following two conditions holds

1) p is prime,
2) $\max_{s\in S}\{s\} \cdot (r-1) < p$,

then for all $s_1 \neq s_2 \in S$ and $t_1, t_2 \in [p]$, $|D_{s_1,t_1} \cap D_{s_2,t_2}| \leqslant 1$.

Now we are ready to prove that the Array Construction can result in k-PIR codes.

Theorem 38: For all $r \leqslant p$ and $S \subseteq [p]$, such that $|S| = k-1 \leqslant p$. If one of the following two conditions holds

1) p is prime,
2) $\max_{s\in S}\{s\} \cdot (r-1) < p$,

then the code $\mathcal{C}_A(r, p, S)$ is an $[rp+(k-1)p, rp, k]^P$ PIR code with rate of $\frac{r}{r+k-1}$. Furthermore, the size of each recovering set is at most r.

Proof: Let $\boldsymbol{x} = (x_{i,j})_{(i,j)\in[r]\times[p]}$ and $E_{r,p,S}(\boldsymbol{x}) = (\rho_{\ell,t})_{(\ell,t)\in[k-1]\times[p]}$, and assume that the set $S = \{s_0, s_1, \ldots, s_{k-2}\}$ is given in an increasing order. Let $(i, j) \in [r] \times [p]$. We show how to construct k recovering sets for $x_{i,j}$. According to Lemma 36, for every $\ell \in [k-1]$, the partition set $P_{s_\ell}(r, p)$ has a set that contains the entry (i, j). Thus, there exist $k-1$ sets $D_{s_0,t_0}, D_{s_1,t_1}, \ldots, D_{s_{k-2},t_{k-2}}$ that each contains the entry (i, j). Furthermore, according to Lemma 37, these sets intersect only on the entry (i, j). Therefore, we conclude that the following $k-1$ sets

$$R_\ell = \{\rho_{\ell,t_\ell}\} \cup \{x_{i',j'} \colon (i', j') \in D_{s_\ell,t_\ell} \setminus \{(i, j)\}\},$$

for $\ell \in [k-1]$, are $k-1$ mutually disjoint recovering sets for $x_{i,j}$. The last recovery set is $\{x_{i,j}\}$, which does not intersect with any of the other previously chosen sets. It is also straightforward to verify that the size of each set is at most r, and that the code rate is $\frac{r}{r+k-1}$. ∎

Note that the Array Construction results also with binary LRCs with availability, i.e., (r, k)-PIR codes, however it does not necessarily improve upon previous results, and anyway our focus here is only for PIR codes. Furthermore, since PIR codes do not require recovering sets of bounded size, we can use the previous construction with large value of r in order to minimize the redundancy.

Corollary 39: Let $n = p^2$, where p is a prime number, and $k \leqslant \sqrt{n}$. The code $\mathcal{C}_A(r = p, p, S = [k-1])$ is a k-PIR code with redundancy $(k-1)\sqrt{n}$. In particular, for all $k \leqslant \sqrt{n}$, $r_P(n, k) = \mathcal{O}(k\sqrt{n})$.

It can be shown using the DTI codes from [17] that the result $r_P(n, k) = \mathcal{O}(k\sqrt{n})$ holds for all $k = n^\epsilon$, where $0 < \epsilon < 1$. However, to the best of our knowledge, the redundancy of $k\sqrt{n}$ was not achieved when k is any other arbitrary function of n, for example $k = \log(n)$, and hence this construction closes on this gap as well.

Remark 40: It shall be noted that the partition $P_s(p, p)$ generates a latin square (see [14], Chapter 1). A latin square is a square matrix with p^2 entries using p different elements, none of them occurring twice within any row or column of the matrix. Moreover, the partitions $P_0(p, p), P_1(p, p), \ldots, P_{p-1}(p, p)$ are in fact mutually orthogonal latin squares (see [14, Ch. 5]). Finally, since latin squares have the attributes required by the array construction, the array construction can be defined based on general latin squares rather than those specified in this paper, and a set of mutually orthogonal latin squares can result in PIR codes.

VIII. The Array Construction for Batch Codes

In this section, we will study how to apply the Array Construction for batch codes. In the previous section, we noticed

that even though the Array Construction can result in constructions of LRCs with availability, it does not necessary result with higher rates than existing ones when r is a constant [20], [22], [32]. However, this construction can result with good batch codes as well as batch codes with restricted size for the recovering sets.

A. A Framework to Construct Batch Codes From the Array Construction

In this section, we show how to construct batch codes using the Array Construction. The idea here is to choose the set S in a way that for every bit, each of its recovering sets intersects with at most one recovering set of any other bit. This property for constructing batch codes from PIR codes was proved in [23] and is stated below.

Lemma 41 [23]: Let $\mathcal{C}$ be an (r,k)-PIR code. Assume that for every two distinct indices $i, j \in [n]$, it holds that each recovering set of the ith bit intersects with at most one recovering set of the jth bit. Then, the code $\mathcal{C}$ is an (r,k)-batch code.

From the above lemma, it is readily verified that the code $\mathcal{C}_A(2, n/2, S=[k])$ is an $(r=2,k)$-batch code with rate $\frac{2}{2+k}$. Next we show how to construct (r,k)-batch codes for arbitrary $r>2$. The main challenge is to find sets S that will generate recovering sets which satisfy the condition in Lemma 41. For that, we use the following definition.

*Definition 42: Let r be a positive integer, and S be a set of non-negative integers. We say that the set S does not contain an **r-weighted arithmetic progression*** if there do* not *exist $s_1, s_2, s_3 \in S$ and $0 < x, y < r-1$, where $x+y<r$, such that*

$$xs_1 + ys_2 = (x+y)s_3 \tag{4}$$

*We say that the set S does not contain an **r-weighted arithmetic progression modulo** p if equation (4) does not hold modulo p.*

Given this definition, we prove the following theorem.

Theorem 43: Let $\mathcal{C}_A(r,p,S)$ be a code from the Array Construction, where $r \leqslant p$ and $S \subseteq [p]$, $|S| = k \leqslant p$. If one of the following two conditions holds

- *p is prime, and S does not contain an r-weighted arithmetic progression modulo p,*
- *$\max_{s\in S}\{s\}\cdot(r-1) < p$, S does not contain an r-weighted arithmetic progression,*

then the code $\mathcal{C}_A(r,p,S)$ is an (r,k)-batch code of dimension rp and redundancy kp.

Proof: Assume that the set $S = \{s_0, s_1, \ldots, s_{k-1}\}$ is in an increasing order. According to Theorem 38, the code $\mathcal{C}_A(r,p,S)$ is an $(r,k+1)$-PIR code. Therefore, it remains to prove that the code $\mathcal{C}_A(r,p,S)$ satisfies the condition of Lemma 41 with respect to the first k recovery sets of every bit. Every bit in position $(i,j) \in [r]\times[p]$ has k mutually disjoint recovering sets whose structure is described in the proof of Theorem 38. In particular, for $\ell \in [k]$, the ℓth recovering set of (i,j) is

$$R_\ell^{(i,j)} = \{\rho_{\ell,t_\ell}\} \cup \{x_{i',j'} :\ (i',j') \in D_{s_\ell,t_\ell} \setminus \{(i,j)\}\},$$

for some $t_\ell \in [p]$, and we denote $D(R_\ell^{(i,j)}) = D_{s_\ell,t_\ell}$. Assume in the contrary that there exist two bits in positions $(i,j),(i',j') \in [r]\times[p]$ such the bit (i,j) has a recovering set $R_{\ell_1}^{(i,j)}$ that intersects with two recovering sets $R_{\ell'_1}^{(i',j')}, R_{\ell'_2}^{(i',j')}$ of (i',j'). Assume that $b_1 \in R_{\ell_1}^{(i,j)} \cap R_{\ell'_1}^{(i',j')}$ and $b_2 \in R_{\ell_1}^{(i,j)} \cap R_{\ell'_2}^{(i',j')}$ where b_1, b_2 are codeword entries. Assume that

$$\begin{aligned} D(R_{\ell_1}^{(i,j)}) &= D_{s'_1,t_1} \\ D(R_{\ell'_1}^{(i',j')}) &= D_{s'_2,t_2} \\ D(R_{\ell'_2}^{(i',j')}) &= D_{s'_3,t_3} \end{aligned}$$

for some $s'_1, s'_2, s'_3 \in S$ and $t_1, t_2, t_3 \in [p]$. Since $(i',j') \in D_{s'_2,t_2} \cap D_{s'_3,t_3}$, we deduce by Lemma 37 that $D_{s'_2,t_2} \cap D_{s'_3,t_3} = \{(i',j')\}$. If b_1 corresponds to a parity bit, then $D_{s'_1,t_1} = D_{s'_2,t_2}$, and then $s'_1 = s'_2$ which cannot happen since $\big(D_{s'_2,t_2} \setminus \{(i,j)\}\big) \cap \big(D_{s'_3,t_3} \setminus \{(i',j')\}\big) = \emptyset$. Therefore, we assume that $b_1 = x_{i_1,j_1}, b_2 = x_{i_2,j_2}$, for some $(i_1,j_1),(i_2,j_2) \in [r]\times[p]$. Thus we get that

$$\begin{aligned} &(i_1,j_1),(i_2,j_2) \in D_{s'_1,t_1}, \\ &(i_1,j_1),(i',j') \in D_{s'_2,t_2}, \\ &(i_2,j_2),(i',j') \in D_{s'_3,t_3}. \end{aligned}$$

We know that $s'_1 \neq s'_2 \neq s'_3$ since these sets intersect. In addition, since these elements appear together in sets of some partition, we deduce that $i' \neq i_1 \neq i_2$. Assume w.l.o.g. $i_1 < i_2 < i'$. It follows that:

$$\begin{aligned} j_1 &= \langle t_1 + i_1 s'_1\rangle_p \quad j_2 = \langle t_1 + i_2 s'_1\rangle_p \\ j_1 &= \langle t_2 + i_1 s'_2\rangle_p \quad j' = \langle t_2 + i' s'_2\rangle_p \\ j_2 &= \langle t_3 + i_2 s'_3\rangle_p \quad j' = \langle t_3 + i' s'_3\rangle_p \end{aligned}$$

This implies that

$$\begin{aligned} \langle j_2 - j_1\rangle_p &= \langle (i_2 - i_1)s'_1\rangle_p \\ \langle j' - j_1\rangle_p &= \langle (i' - i_1)s'_2\rangle_p \\ \langle j' - j_2\rangle_p &= \langle (i' - i_2)s'_3\rangle_p \end{aligned}$$

and therefore

$$\langle (i_2 - i_1)s'_1 + (i' - i_2)s'_3\rangle_p = \langle (i' - i_1)s'_2\rangle_p.$$

Denote $x = i_2 - i_1$, $y = i' - i_2$. We get

$$\langle xs'_1 + ys'_2\rangle_p = \langle (x+y)s'_3\rangle_p, \tag{5}$$

where $s'_1, s'_2, s'_3 \in S$ and $0 < x, y < r-1, 0 < x+y < r$. If the first condition holds, then we get a contradiction to the fact that the set S does not contain an r-weighted arithmetic progression modulo p. If the second condition holds, then $(x+y)s'_3 < p$ and $xs'_1 + ys'_2 \leqslant \max_{s\in S}\{s\}\cdot(r-1) < p$, therefore $xs'_1 + ys'_2 = (x+y)s'_3$, which contradicts the fact that S does not contain an r-weighted arithmetic progression. ∎

In order to complete the construction of batch codes, we are left with the problem of finding large sets S which satisfy one of the two conditions in Theorem 43. That is, given r and p,

our goal is to find the largest such a set S. For example, one can verify that the set $S = \{2^0, 2^\ell, 2^{2\ell}, \ldots, 2^{(k-1)\ell}\}$ where $\ell = \lceil \log(r) \rceil + 1$ and $k \leqslant \frac{\log(p/r)}{\ell} + 1$ satisfies the second condition and does not contain an r-weighted arithmetic progression. However, the achievable value of k using this set will be too small. In the next subsection we will find sets that satisfy the first condition in Theorem 43 when r is not necessarily fixed, and thus they will generate constructions of k-batch codes. Then, in the following subsection, we will study sets that satisfy the second condition for fixed r in order to construct (r, k)-batch codes.

B. Construction of k-Batch Codes

In this section we present an algorithm, called *the Greedy Algorithm*, that generates sets S that satisfy the first condition in Theorem 43 for any values of r and p. Given r and p, the set S is constructed by adding non-negative integers as long as they do not cause an r-weighted arithmetic progression modulo p. The algorithm terminates when it is no longer possible to add numbers to the set S, and then we denote its output by the set $S = \{s_0, s_1, \ldots, s_{k-1}\}$.

Algorithm 1 The Greedy Algorithm

1: **Initialize:**
   $S \leftarrow \{0, 1\}$, $i \leftarrow 2$, $num \leftarrow 2$
2: **while** $num < p$ **do**
3: **if** $S \cup \{num\}$ does not contain an r-weighted arithmetic progression modulo p **then**
4: $S \leftarrow S \cup \{num\}$
5: $i \leftarrow i + 1$
6: $num \leftarrow num + 1$
7: **return** S

The correctness and the properties of the algorithms are proved in the next theorem.

Theorem 44: Let r, p be positive integers, such that p is prime. Then the output of the Greedy Algorithm is a set S of size at least k, where k is the largest integer such that $p > 2k^2r^2$.

Proof: We prove that if $|S| < k$ then the algorithm can add more elements to S. Assume that $S = \{s_0, s_1, \ldots, s_{i-1}\}$ where $i < k$. A number s' cannot be added to S if there exist $0 < x, y < r - 1$ such that $x + y < r$, and $s'_1, s'_2 \in S$ such that

$$\langle xs'_1 + ys'_2 \rangle_p = \langle (x + y)s' \rangle_p \text{ or } \langle xs' + ys'_2 \rangle_p = \langle (x + y)s'_1 \rangle_p.$$

This means that the number of bad elements (elements which cannot be added to S) is at most $2i^2r^2 < 2k^2r^2$, thus proving the claim. ∎

The following theorem follows from these observations.

Theorem 45: For every r, k, let $n = rp$, where p is the smallest prime number such that $2k^2r^2 < p$. Then, there exists an (r, k)-batch code of dimension n and rate $\frac{r}{r+k}$. In particular, the redundancy of the code equals kp.

According to Theorem 45 we are now at a point to construct k-batch codes with good redundancy when k is a function of n.

Corollary 46: For any n and k such that $k = o(\sqrt{n})$, there exists a k-batch code of dimension n and redundancy $\mathcal{O}(n^{\frac{2}{3}}k^{\frac{5}{3}})$. In particular, for $0 < \epsilon < 1/2$, it holds that

$$r_B(n, n^\epsilon) = \mathcal{O}(n^{2/3+5\epsilon/3}).$$

Proof: For n and k, let us choose $r = \lceil n^{\frac{1}{3}}/k^{\frac{2}{3}} \rceil$, and p is the smallest prime number such that $2k^2r^2 < p$. Then, according to Theorem 45, there exists an (r, k)-batch code of dimension $pr > n$ and redundancy kp. That is, the redundancy satisfies

$$kp = \Theta(k^3r^2) = \Theta(n^{\frac{2}{3}}k^{\frac{5}{3}}).$$

The second statement in the corollary is established for $k = n^\epsilon$ in the last equation. ∎

C. Construction of (r, k)-Batch Codes

In this part we construct sets that satisfy the second condition in Theorem 43 for fixed values of r. That is, we construct (r, k)-batch codes using sets S of size k that do not contain r-weighted arithmetic progressions. As before, in order to achieve large value of k, we seek to minimize the maximum number in the set S.

For the specific case of $r = 3$, it holds that containing a 3-weighted arithmetic progression is equivalent to the problem of containing 3-*term arithmetic progression*, which was well studied in the literature. Erdös and Turan [10] initiated the study of sequences that do not contain three terms in arithmetical progression,[2] i.e., a sequence $a_0, a_1, \ldots, a_{k-1}$ such that for any $i \neq j \neq \ell \in [k]$ the equation $a_i + a_j = 2a_\ell$ does not hold. Behrend [2] shows that for every $0 < \alpha < 1$, and sufficiently large p, there exists a set $S \subseteq [p]$ of size $\Omega(p^\alpha)$ such that S does not contain a 3-term arithmetic progression. We generalize the construction of Behrend for any fixed value of r. In particular, we show in Theorem 60 in Appendix E that for any fixed r there exists a set $R \subseteq [p]$ that does not contain an r-weighted arithmetic progression such that $|R| = \Omega(p^\alpha)$. Using these sets to construct batch codes results with the following theorem, which is proved in Appendix E.

Theorem 47: For every $0 < \alpha < 1$, $k = \mathcal{O}(n^\alpha)$, and fixed $r \geqslant 3$, there exists an (r, k)-batch code of dimension n and rate $\frac{r}{r+k}$.

IX. Improved Construction of Batch Codes From the Array Construction

In this section, we provide a generalization for the array construction, which results in the construction of non-binary k-batch codes of redundancy $\Theta(\sqrt{n})$ for any fixed k. In the original array construction, every diagonal was encoded by adding a parity bit which equals the sum of all the bits in the diagonal. In other words, every diagonal was encoded using the simple parity code of minimum distance two. Furthermore, we had to pick the set of diagonals carefully so that the code will be a batch code. Here we show that by using codes of distance $d = k$ to encode the diagonals, we can use all the

[2] in fact, they called them *A sequences* but subsequent papers used the term *without* 3-*term arithmetic progression*.

diagonals in the array and the resulting code will still be a batch code.

Construction 48 (Generalized Array Construction): Let $\mathbb{F}$ be a finite field of size q, and let r, p, n, k be positive integers such that $k \leqslant p$, $n = rp$, and $S \subseteq [p]$ be a subset of size k. We denote $S = \{s_0, s_1, \ldots, s_{k-1}\}$ where $0 \leqslant s_0 < s_1 < \cdots < s_{k-1} \leqslant p-1$. Let $\varepsilon : \mathbb{F}^r \to \mathbb{F}^{r+m}$ be the systematic encoder of an $[r+m, r]_q$ code, i.e., of dimension r and redundancy m over $\mathbb{F}$, so we can write $\varepsilon(x_1, x_2, \ldots, x_r) = (x_1, x_2, \ldots, x_r, \rho_1, \rho_2, \ldots, \rho_m)$ and for simplicity we consider only the redundancy part of the encoder, so $\varepsilon(x_1, x_2, \ldots, x_r) = (\rho_1, \rho_2, \ldots, \rho_m)$. We define the encoder $E_{r,p,S,\varepsilon}$, as a mapping $E_{r,p,S,\varepsilon} : \mathbb{F}^n \to \mathbb{F}^{k \cdot p \cdot m}$ as follows. The input vector $\boldsymbol{x} \in \mathbb{F}^n$ is represented as an $r \times p$ array, that is $\boldsymbol{x} = (x_{i,j})_{(i,j)\in[r]\times[p]}$ and is encoded to the following $k \cdot p$ redundancy words $\rho_{\ell,t} \in \mathbb{F}^m$ of size m, for $\ell \in [k]$, and $t \in [p]$,

$$\rho_{\ell,t} = \varepsilon(\boldsymbol{x}_{\ell,t}), \quad \text{where } \boldsymbol{x}_{\ell,t} = (x_{i,j})_{(i,j)\in D_{s_\ell,t}}.$$

Let $E_{r,p,S,\varepsilon}(\boldsymbol{x}) = (\rho_{0,0}, \ldots, \rho_{0,p-1}, \ldots, \rho_{k-1,0}, \ldots, \rho_{k-1,p-1})$, and the code $\mathcal{C}_G(r, p, S, \varepsilon)$ is defined to be

$$\mathcal{C}_G(r, p, S, \varepsilon) = \{(\boldsymbol{x}, E_{r,p,S,\varepsilon}(\boldsymbol{x})) \colon \boldsymbol{x} \in \mathbb{F}^n\}.$$

In the next lemma we show how to use codes with minimum distance k in the Generalized Array Construction in order to construct k-batch codes.

Lemma 49: Let $\mathbb{F}$ be a finite field, r, p, k be positive integers and $S \subseteq [p]$ be a set of size $k-1 \leqslant p$. Let ε be a systematic encoder of a code of dimension r, minimum distance k and redundancy m over $\mathbb{F}$. If p is a prime number then the code $\mathcal{C} = \mathcal{C}_G(r, p, S, \varepsilon)$ is a k-batch code of dimension $n = rp$ and redundancy $(k-1)pm$ over $\mathbb{F}$.

Proof: Denote $S = \{s_0, s_1, \ldots, s_{k-2}\}$ and let $R = \{(i_0, j_0), (i_1, j_1), \ldots, (i_{k-1}, j_{k-1})\} \subseteq [r] \times [p]$ be the multiset of k requested symbols. We assume w.l.o.g. that the last z symbols in R are different, and all the other $k-z$ symbols are repetitions of these symbols. Denote $w = k - z$, $U = \{(i_w, j_w), \ldots, (i_{k-1}, j_{k-1})\}$, and $P = R \setminus U$. Now we show how to construct k disjoint recovering sets for these symbols. First, for symbols in U, we simply recover them by the trivial recovery sets, i.e., the symbol itself. For symbols inside P, we use properties of the original array construction. According to Lemma 36, for every $\ell \in [w]$, the partition set $P_{s_\ell}(r, p)$ has a set that contains the entry (i_ℓ, j_ℓ). Thus, there exist w sets $D_{s_0,t_0}, D_{s_1,t_1}, \ldots, D_{s_{w-1},t_{w-1}}$ such that $(i_\ell, j_\ell) \in D_{s_\ell,t_\ell}$ for every $\ell \in [w]$.

For $\ell \in [w]$, let I_ℓ denote the set of all the symbols that are contained in D_{s_ℓ,t_ℓ} and another set $D_{s_{\ell'},t_{\ell'}}$ for $\ell' \neq \ell \in [w]$ or U. Formally,

$$I_\ell = \{(i,j) \in D_{s_\ell,t_\ell} \colon \exists \ell' \neq \ell \in [w], (i,j) \in D_{s_{\ell'},t_{\ell'}} \cup U\}.$$

According to Lemma 37, $|D_{s_{\ell_1},t_{\ell_1}} \cap D_{s_{\ell_2},t_{\ell_2}}| \leqslant 1$ for any $\ell_1 \neq \ell_2 \in [k-1]$, and $|U| = k - w$, therefore it follows that $|I_\ell| \leqslant |U| + w - 1 = k - 1$ for any $\ell \in [w]$. Now, for $\ell \in [w]$, we define

$$R_\ell = \{\rho_{\ell,t_\ell}\} \cup \{x_{i,j} \colon (i,j) \in D_{s_\ell,t_\ell} \setminus I_\ell\}.$$

These w sets (together with the previously chosen $k-w$ trivial sets) are disjoint since the intersection between them was removed by removing the symbols in I_ℓ from every set. It remains to prove that R_ℓ is a recovery set for (i_ℓ, j_ℓ) when $\ell \in [w]$. To this end, notice that the vector $\boldsymbol{c} = (\boldsymbol{v}, \rho_{\ell,t_\ell})$, where $\boldsymbol{v} = (x_{i,j} : (i,j) \in D_{s_\ell,t_\ell})$, is a codeword of a code with minimum distance k. Reading the symbols of R_ℓ means we are able to construct the vector $\boldsymbol{c}_e = (\boldsymbol{v}_e, \rho_{\ell,t_\ell})$, where $\boldsymbol{v}_e = (x_{i,j} : (i,j) \in R_\ell)$. Since $|I_\ell| \leqslant k-1$, it follows that $\boldsymbol{c}_e$ can be received from $\boldsymbol{c}$ by erasing at most $k-1$ entries, and therefore $\boldsymbol{c}$ can be recovered from $\boldsymbol{c}_e$. Finally, since $(i_\ell, j_\ell) \in D_{s_\ell,t_\ell}$, we have also recovered the symbol at position (i_ℓ, j_ℓ). ∎

Next, we show an explicit result for a construction of batch codes.

Theorem 50: Let n, p, q be positive integers such that $n = p^2$, p is a prime number and $p + k - 1 < q$. Then for $k \leqslant p+1$, there exists a k-batch code over $\mathbb{F}_q$ of dimension n and redundancy $(k-1)^2 p$.

Proof: The proof follows directly from Lemma 49 by using an MDS code over $\mathbb{F}_q$ of dimension p, minimum distance k, and redundancy $k-1$. ∎

The following corollary follows directly from the last theorem.

Corollary 51: For sufficiently large n, and fixed value of k, there exists a k-batch code of dimension n with redundancy $\Theta(\sqrt{n})$ over $\mathbb{F}_q$, where q satisfies $q > \sqrt{n} + k - 1$.

The next theorem will establish the result for binary batch codes.

Theorem 52: Let n, p be positive integers such that $n = p^2$ and p is a prime number. Then, for $k < p/\log p$, there exists a binary k-batch code of dimension n and redundancy $\mathcal{O}((k-1)^2\sqrt{n}\log(n))$.

Proof: According to Lemma 49, it suffices to find a binary code $\mathcal{C}$ of dimension p, minimum distance k, and redundancy $\mathcal{O}((k-1)\log(p))$. To accomplish this task for $k < p/\log p$, we use BCH codes (see [24, Ch. 8.4]), and therefore the theorem is proved. ∎

The following corollary follows directly from the last theorem.

Corollary 53: For $k \leqslant \sqrt{n}/\log n$, it holds that

$$r_B(n,k) = \mathcal{O}((k-1)^2\sqrt{n}\log(n)).$$

In particular, for $0 < \epsilon \leqslant 0.25$, it holds that

$$r_B(n, k = n^\epsilon) = \mathcal{O}(n^{0.5+2\epsilon^+}),$$

and $r_B(n,k) = \mathcal{O}(\sqrt{n}\log(n))$ for fixed k.

It should be noted that [27] constructed binary k-batch codes with redundancy $\mathcal{O}(\sqrt{n}\log(n))$ for fixed k. However, the constant multiplying $\sqrt{n}\log(n)$ in their work was significantly larger than our constant. More precisely, we get that $r_B(n,k) = \mathcal{O}(k^2\sqrt{n}\log(n))$ while [27] get that $r_B(n,k) = \mathcal{O}(k^2\ e^{k/2}\sqrt{n}\log(n))$ for fixed k.

Let us denote

$$r_B(k = n^\epsilon) = \mathcal{O}(n^{\delta(\epsilon)}).$$

In Fig. 4 we plot upper and lower bounds on the asymptotic behavior of the redundancy of batch codes, i.e. $\delta(\epsilon)$. The lower

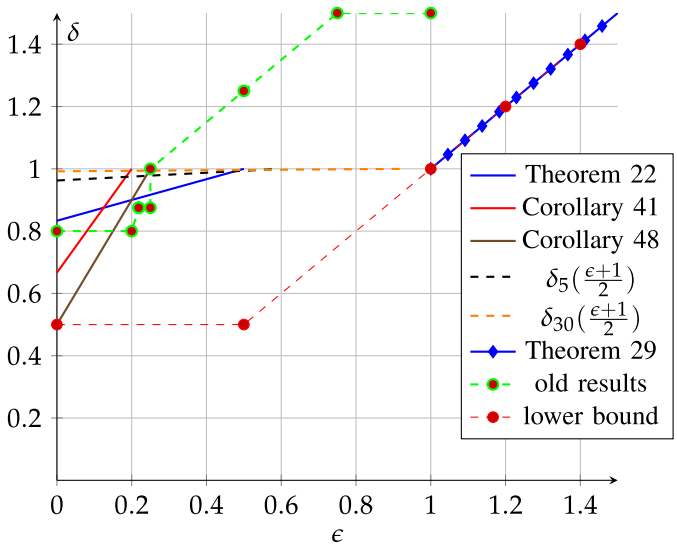

Fig. 4. Asymptotic results for binary batch codes.

bound is the same as PIR codes, while the upper bounds are received from Theorem 25 from Section V, and Corollary 46 from Section VIII, and Corollary 53 from Section IX, and Corollary 31 and Theorem 32 from Section VI. Moreover, note that Reed-Muller codes can be used to construct batch codes for $\epsilon < 1$ using the techniques described in section V. However, since Reed-Muller codes have rate at most $1/2$, their upper bounds were not added to the figure, as they will result with $\delta = 1$ for all $\epsilon < 1$.

X. Other Results

In this section we prove two more results on batch codes. First we show that for $k = 5$, there exists a batch code with redundancy $\theta(\sqrt{n})$, thereby improving a previous result from [27]. Then, we show a similar order of redundancy can be achieved for any fixed k if at the multiset request every bit can be requested at most twice.

We show how to use Construction 34 in order to construct 5-batch codes with optimal order of redundancy. The proofs of Theorem 54 and Theorem 55 are deferred to Appendix F.

Theorem 54: Let $n = p^2$ where p is a prime number. The code $\mathcal{C}$, that extends $\mathcal{C}_A(r = p, p, S = [4])$ by adding one all-parity bit, is a 5-batch code with redundancy $4p+1 = \theta(\sqrt{n})$.

Next we show how to construct k-batch codes with multiplicity 2 in the request set. That is, every bit can be requested at most twice.

Theorem 55: Let $n = p^2$ where p is prime, and let $5 \leqslant k$ be a positive integer. Assume that $\frac{(k-2)(k-3)}{2} < p$. Then, the code $\mathcal{C} = \mathcal{C}_A(r = p, p, S = [\frac{(k-2)(k-3)}{2} + 1]$ is a k-batch code under the constraint that each bit is requested at most twice.

XI. Conclusion

In this work we studied constructions of PIR and batch codes when $k = n^{\epsilon}$ or when the value of k is a fixed constant. For $k = o(n)$, we show that there exist asymptotically optimal k-PIR and k-batch codes with rate approaching one. Fig. 3 summarizes the results for the PIR case when $k = n^{\epsilon}$, while Fig. 4 shows the results for batch codes. When k is fixed, the best constructions for PIR codes achieve redundancy $\mathcal{O}(k\sqrt{n})$. More precisely, PIR codes based on Steiner systems from [11] result in $\sqrt{(k-1)(k-2)n}$, while *Type-1 DTI* codes from [17] achieve redundancy of $(k-2)\sqrt{n}$, only when k is of the form $k = 2^{\ell} + 2$. Our PIR codes in this paper, which are based on the array construction, have redundancy of $(k-1)\sqrt{n}$. For the case of batch codes, we improve upon the previous results for fixed k and construct binary k-batch codes with redundancy of $\mathcal{O}((k-1)^2\sqrt{n}\log n)$, and non-binary k-batch codes with redundancy $\mathcal{O}((k-1)^2\sqrt{n})$, when the field size is at least $\sqrt{n} + k - 1$.

Even though this work does not introduce new results for the lower bound of the codes studied in this paper, we strongly believe that it can be significantly improved. Currently, the best lower bound for k-PIR codes or k-batch codes is $\Omega(\max\{k, \sqrt{n}\})$, which implies that, asymptotically, the lower bound for $k = 3$ is the same as for $k = \sqrt{n}$. Furthermore, despite the fact that batch codes impose a stronger requirement than PIR codes, they still have the same lower bound as PIR codes.

Appendix A

A. Proof of Lemma 10

Assume in the contrary that there exist two different polynomials $P_1, P_2 \in \mathbb{F}_q[x_1, \ldots, x_s]$ of degree at most d that have equal values on the interpolation set A. That is, for $\boldsymbol{a} \in A$, $P_1(\boldsymbol{a}) = P_2(\boldsymbol{a})$. We prove that for every point $\boldsymbol{x} = (x_1, \ldots, x_s) \in \mathbb{F}_q^s$, $P_1(\boldsymbol{x}) = P_2(\boldsymbol{x})$. The proof is by induction on the number of coordinates x_i of $\boldsymbol{x}$ such that $x_i \notin A_i$. Formally, the induction is on $n(\boldsymbol{x}) = |\{i \in [s] : x_i \notin A_i\}|$. The base is trivial since for $\boldsymbol{x}$, such that $n(\boldsymbol{x}) = 0$, $\boldsymbol{x} \in A$ and this holds according to our assumption. Next, we assume that the induction assumption holds, i.e., for all $\boldsymbol{x} \in F_q^s$ such that $n(\boldsymbol{x}) < k$, $P_1(\boldsymbol{x}) = P_2(\boldsymbol{x})$, and prove the claim for all $\boldsymbol{x}$ such that $n(\boldsymbol{x}) = k$. Let $\boldsymbol{x} = (x_1, \ldots, x_s)$ be such that $n(\boldsymbol{x}) = k > 0$. Assume without loss of generality (w.l.o.g) that $x_1 \notin A_1$. We define the following univariate polynomials

$$p_1'(y) = p_1(y, x_2, \ldots, x_s),$$
$$p_2'(y) = p_2(y, x_2, \ldots, x_s).$$

By the induction assumption, we know that $p_1'(y) = p_2'(y)$ for every $y \in A_1$. Since $\deg(p_1'), \deg(p_2') \leqslant d$, we get that $p_1'(y) = p_2'(y)$ fer every y and thus $p_1'(x_1) = p_2'(x_1)$ which completes the proof. ■

B. Proof of Lemma 11

One can show that the polynomial $P(\boldsymbol{x})/x_s^d$ is a polynomial Q in the variables $\frac{x_1}{x_s}, \frac{x_2}{x_s}, \ldots, \frac{x_{s-1}}{x_s}$ with degree at most d. Therefore, according to Lemma 10, the set A is an interpolation set for Q and thus for P. ■

C. Proof of Lemma 13

Let $r = \lfloor \frac{q}{d+1} \rfloor$, and let $A_1, \ldots, A_r$ be r mutually disjoint sets, each of size $d+1$, that partition the first $(d+1)r$ elements of $\mathbb{F}_q$. For every $\boldsymbol{i} = (i_1, \ldots, i_{s-1}) \in [r]^{s-1}$, we denote by $R_{\boldsymbol{i}}$

the set $R_{\boldsymbol{i}} = A_{i_1} \times \cdots \times A_{i_{s-1}} \times \{1\}$. According to Lemma 11, $R_{\boldsymbol{i}}$ is an interpolation set for every $\boldsymbol{i}$. It remains to prove that $R_{\boldsymbol{i}}$ and $R_{\boldsymbol{i}'}$ are disjoint under multiplication for all $\boldsymbol{i} \neq \boldsymbol{i}'$. Let $\boldsymbol{x} = (x_1, \ldots, x_{s-1}, 1) \in R_{\boldsymbol{i}}$ and $\alpha \in \mathbb{F}_q \setminus \{0\}$. If $\alpha \neq 1$ then clearly $\alpha \boldsymbol{x} \notin R_{\boldsymbol{i}'}$. Thus, we need to prove that $\boldsymbol{x} \notin R_{\boldsymbol{i}'}$. Let j be the first index such that $i_j \neq i'_j$. Then $A_{i_j} \cap A_{i'_j} = \emptyset$ and hence for every $\boldsymbol{x} \in R_{\boldsymbol{i}}$, $x_j \in A_{i_j}$ and $x_j \notin A_{i'_j}$, so $\boldsymbol{x} \notin R_{\boldsymbol{i}'}$. ■

APPENDIX B

A. Proof of Theorem 19

According to Theorem 18, there exists a non-binary k'-PIR code $\mathcal{C}'$ of length N' over the field $\mathbb{F}_Q$ with dimension n' and redundancy r' such that

$$\begin{aligned} k' &= \Theta((n')^{(1-\frac{1}{s})(1-\alpha)}), \\ Q &= (n')^{\Theta((n')^{\alpha})}, \\ r' &= \mathcal{O}((n')^{1-\frac{\alpha}{s}}). \end{aligned}$$

We construct a binary PIR code $\mathcal{C}$ from $\mathcal{C}'$ by converting every symbol of $\mathbb{F}_Q$ to $\log(Q) = \Theta((n')^{\alpha}\log(n'))$ bits. Note that this does not change the value of k since every bit has the corresponding recovering sets of the symbol in $\mathbb{F}_Q$ that it belongs to. We show that the code $\mathcal{C}$ is an $[N, n, k]^P$ PIR code with redundancy $r = N - n$ and these parameters satisfy the following properties, where we also use that $n' = \theta(N')$,

$$\begin{aligned} n &= n'\log(Q) = \Theta((n')^{1+\alpha}\log(n')), \\ N &= N'\log(Q) = \Theta((n')^{1+\alpha}\log(n')) = \Theta(n), \\ k &= k' = \Theta((n')^{(1-\frac{1}{s})(1-\alpha)}), \\ r &= r'\log(Q) = \Theta(r'(n')^{\alpha}\log(n')) = \Theta((n')^{1-\frac{\alpha}{s}+\alpha}\log(n')). \end{aligned}$$

Therefore, we get that $n' = \Theta\big((n/\log(n))^{\frac{1}{1+\alpha}}\big)$ and

$$\begin{aligned} k &= \Theta\big((n/\log(n))^{(1-\frac{1}{s})\frac{1-\alpha}{1+\alpha}}\big), \\ r &= \mathcal{O}\big(n^{1-\frac{\alpha}{s(1+\alpha)}}(\log(n))^{\frac{\alpha}{s(1+\alpha)}}\big). \end{aligned}$$

The second part of the claim can be verified by placing $\epsilon = (1-\frac{1}{s})\frac{1-\alpha}{1+\alpha}$, which implies that $\alpha = \frac{s-1-\epsilon s}{s-1+\epsilon s}$. ■

APPENDIX C

A. Proof of Theorem 24

Let $s = 2, m = \lfloor q^{\alpha} \rfloor$, $k = \lfloor q^{1-2\alpha} \rfloor$, and $d = m(q - km - 2)$. Since $km = o(q)$, we get according to Lemma 23 that $\mathcal{C}_M(m, d, s, q)$ is a k-batch code $[q^2, n, k]_Q^B$, where $n = \frac{\binom{d+2}{2}}{\binom{2+m-1}{2}}$ and $Q = q^{\binom{2+m-1}{2}}$. Since the code length is $N = q^2$, we get

$$\begin{aligned} n &= \frac{\binom{d+2}{2}}{\binom{m+1}{2}} = \Theta(N), \\ k &= \Theta(n^{0.5-\alpha}), \\ Q &= q^{\binom{m+1}{2}} = q^{\Theta(q^{2\alpha})} = q^{\Theta(n^{\alpha})} = n^{\Theta(n^{\alpha})}. \end{aligned}$$

The redundancy of this code is

$$\begin{aligned} r &= q^2 - \frac{\binom{d+2}{2}}{\binom{m+2-1}{2}} = q^2 - \frac{(d+1)(d+2)}{m(m+1)} \\ &\leqslant q^2 - \frac{d^2}{m(m+1)} = q^2 - \frac{m^2(q-km-2)^2}{m(m+1)} \\ &= \frac{(m+1)q^2 - m(q^2 - 4q + 4 - 2(q-2)km + k^2m^2)}{m+1} \\ &= \frac{q^2 + 2(q-2)km^2}{m+1} + \frac{4mq - 4m - k^2m^3}{m+1} \\ &= \mathcal{O}(\frac{q^2}{m} + qkm) = \mathcal{O}(q^{2-\alpha}) = \mathcal{O}(n^{1-\frac{\alpha}{2}}). \end{aligned}$$

The second part of the claim can be verified by denoting $\alpha = 0.5 - \epsilon$. Lastly, in case n is not of the form $\binom{d+s}{s}/\binom{s+m-1}{s}$ then we can use the same technique used for PIR codes. ■

B. Proof of Theorem 25

Let $0 < \alpha' < 0.5$. According to Theorem 24, there exists a k'-batch code, $\mathcal{C}'$, over the field $\mathbb{F}_Q$ of length N', dimension n', redundancy r' such that

$$\begin{aligned} k' &= \Theta((n')^{0.5-\alpha'}) \\ r' &= \mathcal{O}((n')^{1-\frac{\alpha'}{2}}) \\ Q &= (n')^{\Theta((n')^{\alpha'})} \end{aligned}$$

As for PIR codes, we construct binary batch code from $\mathcal{C}'$ by converting every symbol of $\mathbb{F}_Q$ to $\log(Q) = \Theta((n')^{\alpha'}\log(n'))$ bits. Let us now calculate the parameters of the new code. Denote the length, dimension of the binary code by N, n, respectively. The redundancy of the code will be $r = N - n$ and k is the availability parameter of the new code. Since $n' = \theta(N')$ it follows that

$$\begin{aligned} N &= N'\log(Q) = \Theta((n')^{1+\alpha'}\log(n')), \\ n &= n'\log(Q) = \Theta((n')^{1+\alpha'}\log(n')), \\ k &= k' = \Theta((n')^{0.5-\alpha'}), \\ r &= r'\log(Q) = \Theta(r'(n')^{\alpha'}\log(n')) = \Theta((n')^{1+\frac{\alpha'}{2}}\log(n')). \end{aligned}$$

Therefore, we get that $n' = \Theta\big((n/\log(n))^{\frac{1}{1+\alpha'}}\big)$ and

$$\begin{aligned} k &= \Theta\big((n/\log(n))^{\frac{0.5-\alpha'}{1+\alpha'}}\big) = \Theta\big((n/\log(n))^{0.5-\frac{3\alpha'}{2(1+\alpha')}}\big) \\ r &= \mathcal{O}\big(n^{1-\frac{\alpha'}{2(1+\alpha')}}(\log(n))^{\frac{\alpha'}{2(1+\alpha')}}\big) \end{aligned}$$

Denote $\alpha = \frac{3\alpha'}{2(1+\alpha')}$. Then we get that $0 < \alpha \leqslant 0.5$ and

$$\begin{aligned} k &= \Theta\big((n/\log(n))^{0.5-\alpha}\big) \\ r &= \mathcal{O}\big(n^{1-\frac{\alpha}{3}}(\log(n))^{\frac{\alpha}{3}}\big) \end{aligned}$$

The second part of the claim can be verified by denoting $\alpha = 0.5 - \epsilon$. ■

APPENDIX D

A. Proof of Lemma 36

Since the set $P_s(r,p)$ contains p sets, each of size r, it suffices to prove that all the sets in $P_s(r,p)$ are mutually disjoint. Assume in the contrary that there exists $0 \leqslant t_1 < t_2 \leqslant p-1$ such that $D_{s,t_1} \cap D_{s,t_2} \neq \emptyset$. Hence there exist $0 \leqslant i_1, i_2 \leqslant r-1$ such that

$$(i_1, \langle t_1 + i_1 s \rangle_p) = (i_2, \langle t_2 + i_2 s \rangle_p),$$

which implies that $i_1 = i_2$ and $t_1 = t_2$, in contradiction. ■

B. Proof of Lemma 37

Let $s_1 \neq s_2 \in S$ and $t_1, t_2 \in [p]$, and assume in the contrary that there exist two entries $a, b \in [r] \times [p]$, $a \neq b$ that appear in the sets D_{s_1,t_1} and D_{s_2,t_2}. Let $a = (i_1, j_1), b = (i_2, j_2)$ where $i_1, i_2 \in [r]$ and $j_1, j_2 \in [p]$. It is clear that $i_1 \neq i_2$, otherwise a and b cannot appear in the same set. Assume w.l.o.g. that $i_1 < i_2$. Since $a, b \in D_{s_1,t_1}$, we get

$$j_1 = \langle t_1 + i_1 s_1 \rangle_p, \quad j_2 = \langle t_1 + i_2 s_1 \rangle_p.$$

Similarly $a, b \in D_{s_2,t_2}$, therefore

$$j_1 = \langle t_2 + i_1 s_2 \rangle_p, \quad j_2 = \langle t_2 + i_2 s_2 \rangle_p.$$

Thus, we get that

$$\langle (i_2 - i_1) s_1 \rangle_p = \langle (i_2 - i_1) s_2 \rangle_p.$$

If the first condition holds, then p is prime and then we get that $s_1 = s_2$, in contradiction. If the second condition holds then $(i_2 - i_1)s_1, (i_2 - i_1)s_2 < p$, and again we get $s_1 = s_2$ in contradiction.

APPENDIX E

In this appendix we prove Theorem 57. First, we state the result of Behrend for $r = 3$, and then we show how to extend it for any fixed r. The following theorem was proved by Behrend in [2].

Theorem 56: For every $0 < \alpha < 1$, and sufficiently large p, there exists a set $S \subseteq [p]$ of size $\Omega(p^\alpha)$ such that S does not contain a 3-term arithmetic progression.

The following corollary follows directly from Theorem 43 and Theorem 56.

Corollary 57: For every $0 < \alpha < 1$ and $k = \mathcal{O}(n^\alpha)$, there exists a $(3,k)$-batch code of dimension n and rate $\frac{3}{3+k}$.

The technique of Behrend can be extended for any constant value of r. This will guarantee that for every $0 < \alpha < 1$, and for sufficiently large p, there exists a set $S \subseteq [p]$ of size $\Omega(p^\alpha)$ such that S does not contain an r-weighted arithmetic progression. First, we define $v_r(p)$ to be the size of the largest subset of $[p]$ which does not contain an r-weighted arithmetic progression.

Definition 58: For any integers r, $d \geqslant 2$, $n \geqslant 2$, $k \leqslant n(d-1)^2$, we define the set

$$R_{r,k}(n,d) = \{a_1 + a_2(rd-1) + \cdots + a_n(rd-1)^{n-1} : 0 \leqslant a_i < d, a_1^2 + a_2^2 + \cdots + a_n^2 = k\}.$$

Lemma 59: The set $R = R_{r,k}(n,d)$ does not contain an r-weighted arithmetic progression.

Proof: Assume in the contrary that there exist three elements $A_1 \neq A_2 \neq A_3 \in R$ and $0 < x, y < r$ where $x + y < r$ such that $xA_1 + yA_2 = (x+y)A_3$. For a nonnegative integer $A < (rd-1)^n$, we define $u(A)$ to be the unique vector $(a_1, a_2, \ldots, a_n)$ such that $A = a_1 + a_2(rd-1) + \cdots + a_n(rd-1)^{n-1}$ and $0 \leqslant a_i < rd-1$ for $1 \leqslant i \leqslant n$. We define $norm(A)$ to be the Euclidean norm of $u(A)$, i.e., $norm(A) = ||u(A)|| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$. Then,

$$norm(xA_1 + yA_2) = norm((x+y)A_3) = (x+y)\sqrt{k},$$
$$norm(xA_1) + norm(yA_2) = (x+y)\sqrt{k}.$$

Since $A_1, A_2 \in R$ and $0 < x + y < r$ we get according to the triangle inequality that

$$\begin{aligned} norm(xA_1 + yA_2) &= ||u(xA_1 + yA_2)|| \\ &= ||xu(A_1) + yu(A_2)|| \\ &\leqslant ||xu(A_1)|| + ||yu(A_2)|| \\ &= norm(xA_1) + norm(yA_2). \end{aligned}$$

Equality holds if and only if $xu(A_1)$ and $yu(A_2)$ are proportional, which means $u(A_1)$ and $u(A_2)$ are proportional, and therefore also A_1 and A_2. Since the norms of A_1 and A_2 are equal, this means that $A_1 = A_2$ which is a contradiction. ■

We are now ready to prove the main result of this section for a lower bound on the value of $v_r(p)$.

Theorem 60: Let r be a positive integer and $0 < \alpha < 1$. Then for all $\epsilon > 0$ and sufficiently large p, $v_r(p) > p^{1 - \frac{3\log r + \epsilon}{\sqrt{\log p}}}$. In particular, for any fixed r there exists a set $R \subseteq [p]$ that does not contain an r-weighted arithmetic progression such that $|R| = \Omega(p^\alpha)$.

Proof: There are d^n different vectors $(a_1, a_2, \ldots, a_n)$ that satisfy $0 \leqslant a_i < d$, and there are $n(d-1)^2 + 1$ different values of k in the definition of $R_{r,k}(n,d)$. Therefore, there exists a value of k such that $R_{r,k}(n,d)$ contains at least $d^n/(n(d-1)^2+1)$. Since all the terms in $R_{r,k}(n,d)$ are smaller than $(rd-1)^n$, we get that

$$v_r((rd-1)^n) \geqslant \frac{d^n}{n(d-1)^2+1} > \frac{d^{n-2}}{n}.$$

Let p be given. Choose $n = \left\lfloor \sqrt{\frac{\log(p)}{\log(r)}} \right\rfloor$, and d such that

$$(rd-1)^n \leqslant p < (rd+r-1)^n.$$

Then,

$$\begin{aligned} v_r(p) \geqslant v_r((rd-1)^n) &> \frac{d^{n-2}}{n} > \frac{(p^{1/n} - (r-1))^{n-2}}{nr^{n-2}} \\ &= \frac{p^{1-2/n}}{nr^{n-2}}(1 - (r-1)p^{-1/n})^{n-2}. \end{aligned}$$

Since $(1 - (r-1)p^{-1/n})^{n-2} \xrightarrow{p \to \infty} 1$, it follows that for sufficiently large p

$$v_r(p) > \frac{p^{1-2/n}}{nr^{n-1}} = p^{1 - \frac{2}{n} - \frac{\log n}{\log p} - \frac{(n-1)\log r}{\log p}} > p^{1 - \frac{3\log r + \epsilon}{\sqrt{\log p}}},$$

for any $\epsilon > 0$. ■

Thus, Theorem 57 follows directly from Theorem 43 and Theorem 60.

APPENDIX F

A. Proof of Theorem 54

Let $\boldsymbol{x} = (x_{i,j})_{(i,j)\in[p]\times[p]}$ be the information bits, which are systematically encoded to $\boldsymbol{y} = (\boldsymbol{x}, \rho_1, \ldots, \rho_{4p+1})$. We denote the requested bits by $R = \{(i_0, j_0), (i_1, j_1), \ldots, (i_4, j_4)\}$. According to Theorem 38, the code $\mathcal{C}$ is a 5-PIR code and thus every bit has 5 recovering sets. We divide the proof to the following cases.

Case 1: Only a single bit is requested more than once. This case can be easily solved according to Lemma 3 from [27].

Case 2: $R = \{(i_0, j_0), (i_0, j_0), (i_1, j_1), (i_1, j_1), (i_2, j_2)\}$. This case follows immediately from Theorem 55.

Case 3: $R = \{(i_0, j_0), (i_0, j_0), (i_0, j_0), (i_1, j_1), (i_1, j_1)\}$. Since $\mathcal{C}$ is a 5-PIR code, every bit has 5 disjoint recovering sets. Let S_0, S_1, S_2, S_3, S_4 be the recovering sets of (i_0, j_0). Since these sets are mutually disjoint, it follows that at least 4 of them do not contain the bit in location (i_1, j_1). Denote w.l.o.g. these sets by S_1, S_2, S_3, S_4. Thus, we take S_1, S_2, S_3 as recovering sets for (i_0, j_0). As for (i_1, j_1), the first recovery set is $\{x_{i_1,j_1}\}$. Let T be the set of all the entries of the codeword $\boldsymbol{y}$. Denote $U = T \setminus \{S_1 \cup S_2 \cup S_3 \cup S_4 \cup \{x_{i_1,j_1}\}\}$. Now we prove that U is a recovery set for (i_1, j_1). Notice that $\sum_{x\in T} x = 0$ because of the global parity bit. Moreover, since S_1, S_2, S_3, S_4 are recovery sets of (i_0, j_0), we get that $\sum_{t=1}^{4}\sum_{x\in S_t} x = 0$. Therefore, if follows that $\sum_{x\in U} x = x_{i_1,j_1}$. Finally, it is clear that all the chosen recovery sets are disjoint, and therefore the claim is proved. ■

B. Proof of Theorem 55

Let $\boldsymbol{x} = (x_{i,j})_{(i,j)\in[p]\times[p]}$ be the information symbols. Since for $5 \leqslant k$ it holds that $\frac{(k-2)(k-3)}{2} + 2 \geqslant k$, the code $\mathcal{C}$ is a k-PIR code according to Theorem 38, and therefore if at most one bit is requested twice then this can be easily solved according to Lemma 3 from [27]. Otherwise, let $R = \{(i_0, j_0), (i_1, j_1), \ldots, (i_{r-1}, j_{r-1})\}$ be the requested bits, $r \leqslant k-2$, where some of these bits are requested twice. According to Construction 34 and the proof of Theorem 38, the code has $\ell = \frac{(k-2)(k-3)}{2} + 1$ partitions of $[p] \times [p]$, $P_0, P_1, \ldots, P_{\ell-1}$, where every partition corresponds to a different slop in S. We say that a partition P is bad if there exist two elements in R which appear in the same set in P. Since the number of pairs of elements in R is $\frac{r(r-1)}{2} < \ell$, and every such pair appears in at most one set according to Lemma 37, we can find a partition, w.l.o.g. P_0, that is not bad, i.e., each bit in R appears in different set in P_0. Therefore, there exist sets $D_{0,t_0}, D_{0,t_1}, \ldots, D_{0,t_{r-1}} \in P_0$, such that $(i_s, j_s) \in D_{0,t_s}$ for every $s \in [r]$. Furthermore, according to Lemma 36, these sets do not intersect. Therefore, we conclude that the following $2r$ sets

$$R_{\ell_1} = \{\rho_{0,t_\ell}\} \cup \{x_{i',j'} \colon\ (i', j') \in D_{0,t_\ell} \setminus \{(i, j)\}\}$$
$$R_{\ell_2} = \{x_{i_\ell, j_\ell}\}$$

for $\ell \in [r]$, are mutually disjoint recovering sets for the requested bits. ■

ACKNOWLEDGEMENT

The authors thank Alexander Vardy and Ron Roth for helpful discussion as well as Shachar Lovett for his contribution to Section IX. They also thank the three anonymous reviewers and the Associate Editor Prof. Joerg Kliewer for their valuable comments and suggestions, which have contributed for the clarity of the paper and its presentation.

REFERENCES

[1] H. Asi and E. Yaakobi, "Nearly optimal constructions of PIR and batch codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, Germany, Jun. 2017, pp. 151–155.
[2] F. A. Behrend, "On sets of integers which contain no three terms in arithmetical progression," *Proc. Nat. Acad. Sci. USA*, vol. 32, no. 12, pp. 331–332, 1946.
[3] S. Bhattacharya, S. Ruj, and B. Roy, "Combinatorial batch codes: A lower bound and optimal constructions," *Adv. Math. Commun.*, vol. 6, no. 2, pp. 165–174, 2012.
[4] M. Blaum and R. M. Roth, "New array codes for multiple phased burst correction," *IEEE Trans. Inf. Theory*, vol. 39, no. 1, pp. 66–77, Jan. 1993.
[5] R. A. Brualdi, K. P. Kiernan, S. A. Meyer, and M. W. Schroeder, "Combinatorial batch codes and transversal matroids," *Adv. Math. Commun.*, vol. 4, no. 3, pp. 419–431, 2010.
[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] C. Bujtás and Z. Tuza, "Relaxations of Hall's condition: Optimal batch codes with multiple queries," *Appl. Anal. Discrete Math.*, vol. 6, no. 1, pp. 72–81, 2012.
[8] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, 1998.
[9] Y. M. Chee, F. Gao, S. T. H. Teo, and H. Zhang, "Combinatorial systematic switch codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, Jun. 2015, pp. 241–245.
[10] P. Erdös and P. Turan, "On some sequences of integers," *J. London Math. Soc.*, vol. s1-11, pp. 261–264, Oct. 1936.
[11] A. Fazeli, A. Vardy, and E. Yaakobi. (May 2015). "PIR with low storage overhead: Coding instead of replication." [Online]. Available: https://arxiv.org/abs/1505.06241
[12] S. L. Frank-Fischer, V. Guruswami, and M. Wootters. (Apr. 2017). "Locality via partially lifted codes." [Online]. Available: https://arxiv.org/abs/1704.08627
[13] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," in *Proc. 36th Annu. ACM Symp. Theory Comput.*, 2004, pp. 262–271.
[14] A. D. Keedwell and J. Dénes, *Latin Squares and Their Applications*. Amsterdam, The Netherlands: Elsevier, 2015.
[15] S. Kopparty, S. Saraf, and S. Yekhanin, "High-rate codes with sublinear-time decoding," in *Proc. 43rd Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 2011, pp. 167–176.
[16] H.-Y. Lin and E. Rosnes. (Jul. 2017). "Lengthening and extending binary private information retrieval codes." [Online]. Available: https://arxiv.org/abs/1707.03495
[17] S. Lin and D. J. Costello, *Error Control Coding*. Upper Saddle River, NJ, USA: Prentice-Hall, 2004.
[18] H. Lipmaa and V. Skachek, "Linear batch codes," in *Coding Theory and Applications* (CIM Series in Mathematical Sciences), vol. 3. Cham, Switzerland: Springer, 2015, pp. 245–253.
[19] J. L. Massey, *Threshold Decoding*. Cambridge, MA, USA: MIT Press, 1963.
[20] L. Pamies-Juarez, H. D. L. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," in *Proc. IEEE Int. Symp. Inf. Theory*, Istanbul, Turkey, Jul. 2013, pp. 892–896.
[21] S. Rao and A. Vardy. (May 2016). "Lower bound on the redundancy of PIR codes." [Online]. Available: https://arxiv.org/abs/1605.01869
[22] 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./Jul. 2014, pp. 681–685.
[23] A. S. Rawat, Z. Song, A. G. Dimakis, and A. Gál, "Batch codes through dense graphs without short cycles," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1592–1604, Apr. 2016.
[24] R. M. Roth, *Introduction to Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[25] N. Silberstein and A. Gál, "Optimal combinatorial batch codes based on block designs," *Des. Codes Cryptogr.*, vol. 78, pp. 409–424, Feb. 2014.
[26] 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.
[27] A. Vardy and E. Yaakobi, "Constructions of batch codes with near-optimal redundancy," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 1197–1201.
[28] 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*, Aachen, Germany, Jun. 2017, pp. 1421–1425.
[29] Z. Wang, H. M. Kiah, and Y. Cassuto, "Optimal binary switch codes with small query size," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, Jun. 2015, pp. 636–640.
[30] 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.
[31] Z. Wang, O. Shaked, Y. Cassuto, and J. Bruck, "Codes for network switches," in *Proc. IEEE Int. Symp. Inf. Theory*, Istanbul, Turkey, Jul. 2013, pp. 1057–1061.
[32] A. Wang, Z. Zhang, and M. Liu, "Achieving arbitrary locality and availability in binary codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, Jun. 2015, pp. 1866–1870.
[33] M. Wootters, "Linear codes with disjoint repair groups," unpublished manuscript.
[34] S. Yekhanin, "Locally decodable codes," *Found. Trends Theor. Comput. Sci.*, vol. 6, no. 3, pp. 139–255, 2012.
[35] H. Zhang and V. Skachek, "Bounds for batch codes with restricted query size," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 1192–1196.

Hilal Asi was born in Israel in 1994. He received the B.Sc. and M.Sc. degrees from the Technion — Israel Institute of Technology, Haifa, Israel, in 2015 and 2017 respectively, from the Computer Science Department.

He is currently working toward the Ph.D. degree in electrical engineering at Stanford University. His research interests include algebraic error correction coding, coding theory, and their applications for distributed storage.

Eitan Yaakobi (S'07–M'12–SM'17) is an Assistant Professor at the Computer Science Department at the Technion — Israel Institute of Technology. He received the B.A. degrees 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, San Diego, in 2011. Between 2011-2013, he was a postdoctoral researcher in the department of Electrical Engineering at the California Institute of Technology. His research interests include information and coding theory with applications to non-volatile memories, associative memories, data storage and retrieval, and voting theory. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship in 2010-2011.