Constructions of PIR and Batch Codes for Distributed Storage

Hilal Asi
Constructions of PIR and Batch Codes for Distributed Storage

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Hilal Asi

Submitted to the Senate of the Technion — Israel Institute of Technology
Tishrei 5777 Haifa September 2017
The research thesis was done under the supervision of Prof. Eitan Yaakobi in the Computer Science Department.

Acknowledgments

First, I would like to express my deepest gratitude to Prof. Eitan Yaakobi for his support and guidance throughout the work.

I would also like to thank my friends for an amazing five years at the Technion.

Finally, I would like to thank my parents and family for their unlimited support and encouragement.

The generous financial support of the Technion is gratefully acknowledged.
Contents

Abstract 1

Abbreviations and Notations 2

1 Introduction 3
   1.1 Problem Definition ......................... 3
   1.2 Our Work ................................ 5
   1.3 Outline of the Thesis ....................... 5

2 Background and Previous Results 6
   2.1 Definitions and Preliminaries ................ 6
   2.2 The Subcube Construction .................... 8
   2.3 PIR Codes from Steiner Systems ............... 8
   2.4 PIR Codes from One-Step Majority Logic-Decodable Codes .... 9
   2.5 PIR Codes from Partially Lifted Codes ........ 9
   2.6 Lower Bound on the Redundancy of PIR Codes .......... 10
   2.7 Batch Codes from Graphs with High Girth .......... 10
   2.8 Batch Codes from PIR Codes .................. 10
   2.9 Batch Codes from Consecutive Switch Codes ........ 11
   2.10 Summary of Previous Results ................. 11

3 Availability Codes from Multiplicity Codes 14
   3.1 Multiplicity Codes ........................... 14
      3.1.1 Derivatives and Multiplicities .......... 14
      3.1.2 Definition of Multiplicity Codes ....... 15
   3.2 PIR Codes from Multiplicity Codes .......... 17
      3.2.1 Interpolation Sets of Polynomials ...... 17
3.2.2 Construction of Recovering Sets .................................. 19
3.2.3 PIR Codes over the Binary Field ................................ 24
3.3 Batch Codes from Multiplicity Codes .............................. 28

4 The Array Construction ................................................. 33
4.1 The Array Construction for PIR Codes .......................... 33
4.2 The Array Construction for Batch Codes ....................... 39
  4.2.1 A Framework to Construct Batch Codes from the Array
       Construction ..................................................... 39
  4.2.2 Construction of $k$-Batch Codes ............................... 42
  4.2.3 Construction of $(r,k)$-Batch Codes .......................... 44
4.3 Improved Construction of Batch Codes from the Array
    Construction ....................................................... 46
4.4 Construction of Batch Codes from PIR Codes ................. 49
4.5 Other Results ...................................................... 51

5 Conclusions and Future Work ....................................... 54
5.1 Summary .......................................................... 54
5.2 Future Work ....................................................... 55
## List of Figures

2.1 Known results for binary PIR codes ........................................... 12  
2.2 Known results for binary batch codes ................................. 13  
3.1 Asymptotic results for non-binary PIR codes ......................... 25  
3.2 Asymptotic results for binary PIR codes ............................... 27  
3.3 Asymptotic results for binary PIR codes ......................... 29  
4.1 The Greedy Algorithm ....................................................... 43  
4.2 Asymptotic results for binary batch codes ......................... 52
Abstract

This work studies 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 asks this property for every multiset request of $k$ information symbols. In this work we study constructions and bounds for both PIR and batch codes.

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.

The results we achieve in this work are based on two constructions. The first uses multiplicity codes which are a generalization of Reed Muller codes and were first presented by Kopparty et al. in [12]. The second construction we propose is a generalization of the subcube construction from [10]. These constructions can be used to construct both PIR and batch codes.

For the case of PIR codes, we present asymptotically optimal constructions with rate approaching one for large values of $k$. Specifically, when $k = n^e$ and $e < 1$, we construct $k$-PIR codes with rate approaching one. If $e > 1$, then our constructions have nearly optimal length.

For the case of batch codes, we present new results for large and fixed values of $k$. If $k$ is fixed, we construct a non-binary $k$-batch codes with optimal order of the redundancy. In addition, when $k = n^e$ and $e < 1$, we construct $k$-batch codes with rate approaching one. If $e > 1$, then our constructions have nearly optimal length. We also present constructions of batch codes with restricted size of recovery sets.
Abbreviations and Notations

w.l.o.g. — without loss of generality
$n$ — code dimension
$N$ — code Length
$k$ — availability parameter (number of recovering sets)
$\mathbb{F}_q$ — finite field of size $q$
$\mathbb{F}_q^n$ — vector space of dimension $n$ over $\mathbb{F}_q$
Chapter 1

Introduction

1.1 Problem Definition

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 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 [8] due to their applicability for private information retrieval in a coded storage system [5]. They are also very similar to one-step majority-logic decodable codes that were studied a while ago by Massey [16] and later by Lin and others [14] 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 [10]. 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 [4, 6, 25–27].

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 \( k \), \( \lim_{n \to \infty} B(n, k)/n = \lim_{n \to \infty} P(n, k)/n = 1 \), [10], 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 \( k \in \{3, 4\} \), \( r_B(n, k) = r_P(n, k) = \Theta(\sqrt{n}) \) [8, 18, 23, 28]. Furthermore, for any other fixed \( k \), \( r_P(n, k) = \Theta(\sqrt{n}) \) [8, 18, 28] and \( r_B(n, k) = O(\sqrt{n \log(n)}) \) [23]. One of the problems we study in the 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 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 \( k = \Theta(n^\epsilon) \).
1.2 Our Work

The results achieved in the 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. in [12]. These codes were also used for the construction of locally decodable codes. The second construction we used is based on the subcube construction from [10]. This basic construction can be used to construct both PIR and batch codes. While the idea in the works in [8, 10] 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 construction 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 $r_P(n, k), r_B(n, k) \geq k$, this result is indeed optimal.

More results on the value $r_P(n, k), r_B(n, k)$, where $k = \Theta(n^\epsilon)$, are summarized in Fig. 3.3, 4.2, respectively.

1.3 Outline of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, we formally define the codes studied in this paper and review previous results. In Chapter 3, we review multiplicity codes and show how to use them to construct PIR and batch codes. Then, in Chapter 4 we present our array construction and its results for PIR, batch codes, respectively. Furthermore, in Chapter 4.3, we present another extension of the array construction that is invoked to construct more batch codes. In Chapter 4.4, we show how PIR codes can be used for the existence of batch codes. Chapter 4.5 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, Chapter 5 concludes the thesis.
Chapter 2

Background and Previous Results

In this chapter we formally define the codes we study in the thesis, and preview the previous results that are related to our work.

2.1 Definitions and Preliminaries

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 [8] and batch codes that were first studied by Ishai et al. in [10]. 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 \( C \) be an \([N, n]_q\) linear code over the field \( \mathbb{F}_q \).

1. The code \( C \) 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 $C$ will be called a $k$-batch code, and will be denoted by $[N,n,k]^B_q$, 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 [17, 19], 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 [16] and later by Lin and others [14] 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 $[N,n,k]^P_q$, $[N,n,k]^B_q$ code, respectively. Since every batch code is also a PIR code with the same parameters we get that $B(n,k)^q \geq P(n,k)^q$. For the binary case, we will remove $q$ from these and subsequent notations.

**Example 1.** In case $k = 1$, the code $[n,n]^q$ which simply stores all the information symbols is an $[n,n,1]^P_q$ PIR code and also an $[n,n,1]^B_q$ batch code. Hence $P(n,1)^q = B(n,1)^q = n$. Similarly, the simple parity $[n+1,n]^q$ code is an $[n+1,n,2]^P_q$ PIR code and also an $[n+1,n,2]^B_q$ batch code, so $P(n,2)^q = B(n,2)^q = n + 1$.

In [10], it was shown using the subcube construction that for any fixed $k$ there exists an asymptotically optimal construction of $[N,n,k]^B_q$ 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 \( r_B(n, k)_q = B(n, k)_q - n \) and similarly, \( r_P(n, k)_q = P(n, k)_q - n \).

Hence, \( r_B(n, 1)_q = r_P(n, 1)_q = 0 \), \( r_B(n, 2)_q = r_P(n, 2)_q = 1 \).

In this paper, we will mostly study the values of \( r_P(n, k)_q \) and \( r_B(n, k)_q \), 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 \( \epsilon \) for which \( r_P(n, k = \Theta(n^\epsilon)) = o(n) \), and similarly for batch codes.

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 [8, 13, 24] for PIR codes.

### 2.2 The Subcube Construction

In [10], 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

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

In [8], Fazely et al. presented the cubic construction to prove the same result. The advantage is that this construction is not recursive, as it is the case in the subcube construction.

### 2.3 PIR Codes from Steiner Systems

In [8], the authors showed constructions of PIR codes from Steiner systems, from one-step majority logic-decodable codes, from constant-weight codes, and from certain locally recoverable codes. Using these constructions, they proved that for fixed \( k \geq 3 \) it holds that \( r_P(n, k) = \mathcal{O}(\sqrt{n}) \). Together with the lower bound presented in Theorem 7, this shows that the redundancy of \( k \)-PIR codes satisfies \( r_P(n, k) = \Theta(\sqrt{n}) \) for any fixed \( k \geq 3 \). Here we present the results which are based on Steiner systems.

A Steiner system with parameters \( t, \ell, n \), denoted by \( S(t, \ell, n) \), is an \( n \)-element set \( S \) together with a set of \( \ell \)-element subsets of \( S \) (called blocks) with the property that each \( t \)-element subset of \( S \) is contained in exactly one block. Based on Steiner systems, the following result was proved in [8].
Theorem 2. If a Steiner system $S(2, k-1, r)$ exists, then there exists a PIR code $[N, n, k]$ such that $N = \frac{r(r-1)}{(k-1)(k-2)} + r$ and $n = \frac{r(r-1)}{(k-1)(k-2)}$. In particular, it holds that $P(\frac{r(r-1)}{(k-1)(k-2)}, k) \leq \frac{r(r-1)}{(k-1)(k-2)} + r$.

In [8], the authors stated the existence of Steiner systems $S(2, k-1, r)$ for infinitely many values of $r$ when the value of $k$ is fixed. Thus, for large values of $n$, it follows that $r_P(n, k) = O(\sqrt{n})$ for any fixed $k \geq 3$.

2.4 PIR Codes from One-Step Majority Logic-Decodable Codes

Several constructions of one-step majority logic-decodable codes are described in [14]. These constructions can be used to construct PIR codes. The following theorem is based on the DTI codes which are described in Chapter 8 in [14].

Theorem 3. For any positive integers $\theta, \ell$, and $\lambda$, it hold that

$$P(2^{2\theta\ell} - (2^{\theta+1} - 1)^\ell, 2\ell + 2) \leq 2^{2\theta\ell} - 1,$$

$$P((2^\lambda - 1)^\ell, 2^\ell) \leq 2^{3\ell} - 1.$$

The next corollary follows immediately from the last theorem.

Corollary 4. For sufficiently large $n$, and $0 < \epsilon < 1$, it holds that

$$r_P(n, k = n^\epsilon) = O(k\sqrt{n}).$$

The following theorem follows from the Difference Set Codes (see [14], Chapter 8).

Theorem 5. For sufficiently large $n$, it holds that

$$r_P(n, \sqrt{n}) = O(n^{\log_2^3}).$$

2.5 PIR Codes from Partially Lifted Codes

The work of [9] showed how to construct PIR codes from partially lifted codes. They improved upon the result from Corollary 4 when $\epsilon = 0.25$. 

9
Theorem 6. For sufficiently large $n$, it holds that
\[ r_p(n, k = n^{0.25}) = O(n^{0.714}). \]

2.6 Lower Bound on the Redundancy of PIR Codes

Let $[N, n, k]^P$ be a PIR code with redundancy $r = N - n$. If follows immediately that $k - 1 \leq r$ since every non-trivial recovery set (i.e., recovery set which is not the bit itself) must contain at least one parity bit. Recently, Rao and Vardy, and Wootters independently, showed an improvement on the lower bound in [18] and [28] respectively, where they stated that for $3 \leq k$, it holds that $r = \Omega(\sqrt{n})$. More precisely, the following theorem was proved in [18, 28].

Theorem 7. For $k = 3$ and positive integer $n$, it holds that
\[ r_p(n, k)(r_p(n, k) + 1) \geq 2n. \]

Since every $k + 1$-PIR code is also a $k$-PIR code for every $k > 0$, it holds that $r_p(n, k + 1) \geq r_p(n, k)$ for $n \geq 1$. Therefore, the lower bound of Theorem 7 extends from 3-PIR codes to $k$-PIR codes with $3 < k$.

2.7 Batch Codes from Graphs with High Girth

Rawat et al. showed in [20] a method to construct batch codes from dense bipartite graphs with no small cycles. In particular, they showed a construction of batch codes from bipartite graphs with girth (length of the shortest cycle) at least 8.

Theorem 8. For sufficiently large $n$, it holds that
\[
\begin{align*}
    r_B(n, n^{1/3}) &\leq n, \\
    r_B(n, n^\epsilon) &\leq n^{7/8}, \text{ for } 7/32 \leq \epsilon \leq 1/4, \\
    r_B(n, n^\epsilon) &\leq n^{4\epsilon}, \text{ for } 1/5 < \epsilon \leq 7/32.
\end{align*}
\]

2.8 Batch Codes from PIR Codes

In [23], Vardy and Yaakobi showed how to construct batch codes from PIR codes by applying the encoder of the PIR code on different partitions of the input. Their
work focused on batch codes when the value of $k$ is fixed. In particular, they proved the following results for batch codes.

**Lemma 9.** For $k = 3$ or $k = 4$ and positive integer $n$, it holds that

$$r_B(n, k) = r_p(n, k).$$

**Theorem 10.** For $n$ and $k$ large enough

$$r_B(n, k) = k^{2.5} \log(n) e^{k/2} r_p(n, k).$$

In particular, $r_p(n, k) = O(\sqrt{n} \log n)$ for fixed $k$.

### 2.9 Batch Codes from Consecutive Switch Codes

In [4], Buzaglo et al. studied the problem of consecutive switch codes, which are related to batch codes. Their work showed a construction of batch codes when the availability parameter equals the dimension of the code, i.e., $k = n$.

**Theorem 11.** For sufficiently large $n$, it holds that

$$B(n, n) \leq 2n^{1.5}.$$  

### 2.10 Summary of Previous Results

This section summarizes the relevant known results for our problem:

1. $r_p(n, k) = \Theta(\sqrt{n})$ for fixed $k$, [8, 18, 28].
2. $r_p(n, \sqrt{n}) = O(n^{\log 3})$, [14].
3. $r_p(n, n^\epsilon) = O(n^{0.5+\epsilon})$, [14].
4. $r_p(n, n^{0.25}) = O(n^{0.714})$, [9].
5. $r_B(n, k) = O(\sqrt{n} \log n)$ for fixed $k$, [23].
6. $r_B(n, n^{1/3}) \leq n$, [20].
7. $r_B(n, n^\epsilon) \leq n^{7/8}$ for $7/32 \leq \epsilon \leq 1/4$, [20].
8. $r_B(n, n^\epsilon) \leq n^{4\epsilon}$ for $1/5 < \epsilon \leq 7/32$, [20].

9. $B(n, n) \leq 2n^{1.5}$, [4].

If we denote $r_P(n, k = n^\epsilon) = \mathcal{O}(n^{\delta})$ and $r_B(n, k = n^\epsilon) = \mathcal{O}(n^{\delta})$, then Fig. 2.1 and Fig. 2.2 summarize the known results for PIR and batch codes, respectively.
Figure 2.2: Known results for binary batch codes
Chapter 3

Availability Codes from Multiplicity Codes

In this chapter we show how to construct both PIR and batch codes from multiplicity codes. In [12], 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 codeword’s symbol 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.

3.1 Multiplicity Codes

In this section we review the construction of multiplicity codes. This family of codes was first presented by Kopparty et al. in [12] as a generalization of Reed Muller codes by calculating the derivatives of polynomials. We follow the definitions of these codes as were presented in [12] and first start with the definition of the Hasse derivative.

3.1.1 Derivatives and Multiplicities

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

**Definition 12.** For a polynomial $P(x) \in \mathbb{F}[x]$ and a non-negative vector $i$, the $i$-th Hasse derivative of $P(x)$, denoted by $P^{(i)}(x)$, is the coefficient of $z^i$ in the polynomial $P'(x, z) = P(x + z) \in \mathbb{F}[x, z]$.

It is clear from the definition that

$$P(x + z) = \sum_i p^{(i)}(x)z^i.$$

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

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

### 3.1.2 Definition of Multiplicity Codes

**Definition 13.** Let $m, d, s$ be nonnegative integers, $q$ be a prime power, and $i = (i_1, \ldots, i_s)$ be a vector of non-negative integers. Let

$$\Sigma = \mathbb{F}_{q}^{[\text{wt}(i) < m]} = \mathbb{F}_{q}^{s(m-1)}.$$

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 $w \in \mathbb{F}_q^s$, denoted by $P^{(<m)}(w)$, to be the vector

$$P^{(<m)}(w) = \left(p^{(i)}(w)\right)_{\text{wt}(i) < m} \in \Sigma.$$

**Definition 14.** Let $m, d, s$ be nonnegative integers and let $q$ be a prime power. Let

$$\Sigma = \mathbb{F}_{q}^{[\text{wt}(i) < m]} = \mathbb{F}_{q}^{s(m-1)}.$$

The **multiplicity code** $C_M(m, d, s, q)$ 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 $\Sigma$, and has length $q^s$. The coordinates are indexed by elements of $\mathbb{F}_q^s$. For each polynomial $P(x) \in \mathbb{F}_q[x_1, \ldots, x_s]$ with $\text{deg}(P) \leq d$, there is a codeword in $C_M$ given by:

$$\text{Enc}_{m, d, s, q}(P) = \left(P^{(<m)}(w)\right)_{w \in \mathbb{F}_q^s} \in (\Sigma)^{q^s}.$$
That is,
\[ C_M(m, d, s, q) = \left\{ \left( P^{(\leq m)}(w) \right)_{w \in \mathbb{F}_q} \in \Sigma^q: P \in \mathbb{F}_q[x], \deg(P) \leq d \right\}. \]

The following lemma, which calculates the rate and relative distance of multiplicity codes, was proved in [12].

**Lemma 15.** [12, Lemma 9] The multiplicity code \( C_M(m, d, s, q) \) has relative minimum distance at least \( \delta = 1 - \frac{d}{mq} \) and rate \( \frac{(d + s)}{(s + m - 1)}q^s \).

Lastly, we note that since the multiplicity code \( C_M(m, d, s, q) \) 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 Lemma 2.3 in [29]. 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 2.** Let \( q = s = 2, d = 3, \) and \( m = 4 \). Thus we get that
\[ C_M(4, 3, 2, 2) = \{ \left( P^{(\leq 4)}(w) \right)_{w \in \mathbb{F}_2} \in \Sigma^{2^2}: P \in \mathbb{F}_2[x], \deg(P) \leq 3 \}. \]

If we take \( P(x_1, x_2) = x_1 x_2 + x_1 x_2^2 \), then
\[
P(x + z) = P(x_1 + z_1, x_2 + z_2) = (x_1 + z_1)(x_2 + z_2) + (x_1 + z_1)(x_2 + z_2)^2 = x_1 x_2 + x_1 x_2^2 + (x_2 + x_2^2)z_1 + x_1 z_2 + z_1 z_2^2 + x_1 z_2^2 + z_1 z_2^2.
\]

Therefore \( P(x) \) is encoded to \( \left( P^{(\leq 4)}(w) \right)_{w \in \mathbb{F}_2} \) where
\[
\left( P^{(\leq 4)}(w) \right) = (P^{(0,0)}, P^{(1,0)}, P^{(0,1)}, P^{(2,0)}, P^{(1,1)}, P^{(0,2)}, P^{(3,0)}, P^{(2,1)}, P^{(1,2)}, P^{(0,3)}) = (w_1 w_2 + w_1 w_2^2, w_2 + w_2^2, w_1, 0, 1, w_1, 0, 1, 0, 1).
\]
3.2 PIR Codes from Multiplicity Codes

In this section we modify the recovering procedure of symbols in the multiplicity codes 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.

3.2.1 Interpolation Sets of Polynomials

We start with the following definition of interpolating sets.

**Definition 16.** For \( P(x) \in \mathbb{F}_q[x_1, \ldots, x_s] \) and \( R \subseteq \mathbb{F}_q^s \), we say that \( R \) is an interpolation set of \( P(x) \) if for every polynomial \( Q(x) \) such that \( P(x) = Q(x) \) for every \( x \in R \), it holds that \( P(x) = Q(x) \) for every \( x \in \mathbb{F}_q^s \).

Even though the next lemma is somehow basic, we show its proof here for the completeness of the results in the paper.

**Lemma 17.** Let \( P(x) \in \mathbb{F}_q[x_1, \ldots, x_s] \), with \( \deg(P) \leq d \). 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 \( P(x) \).

**Proof** 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 \( a \in A \), \( P_1(a) = P_2(a) \). We prove that for every point \( x = (x_1, \ldots, x_s) \in \mathbb{F}_q^s \), \( P_1(x) = P_2(x) \). The proof is by induction on the number of coordinates \( x_i \) of \( x \) such that \( x_i \notin A_i \). Formally, the induction is on \( n(x) = |\{i \in [s] : x_i \notin A_i\}| \). The base is trivial since for \( x \), such that \( n(x) = 0 \), \( x \in A \) and this holds according to our assumption. Next, we assume that the induction assumption holds, i.e., for all \( x \in \mathbb{F}_q^s \) such that \( n(x) < k \), \( P_1(x) = P_2(x) \), and prove the claim for all \( x \) such that \( n(x) = k \). Let \( x = (x_1, \ldots, x_s) \) be such that \( n(x) = k > 0 \). 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 \( \deg(p_1'), \deg(p_2') \leq d \), we get that \( p_1'(y) = p_2'(y) \) for every \( y \) and thus \( p_1'(x_1) = p_2'(x_1) \) which completes the proof.

According to Lemma 17, we can now generate a large number of disjoint interpolation sets for homogenous polynomials.

**Lemma 18.** Let \( P(x) \in \mathbb{F}_q[x_1, \ldots, x_s] \) 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 \( P(x) \), where \( 1 \in \mathbb{F}_q \) is the unitary element of the field.

**Proof** One can show that the polynomial \( P(x)/x_1^d \) is a polynomial \( Q \) in the variables \( x_2, x_3, \ldots, x_s \) with degree at most \( d \). Therefore, according to Lemma 17, the set \( A \) is an interpolation set for \( Q \) and thus for \( P \).

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

**Definition 19.** 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 20.** Let \( P(x) \in \mathbb{F}_q[x_1, \ldots, x_s] \) be a homogeneous polynomial such that \( \deg(P) = d \). Then there exist \( \left\lfloor \frac{d}{d+1} \right\rfloor^{s-1} \) interpolation sets of \( P(x) \), each of size \( (d+1)^{s-1} \), which are mutually disjoint under multiplication.

**Proof** Let \( r = \left\lfloor \frac{d}{d+1} \right\rfloor \), 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 \( i = (i_1, \ldots, i_{s-1}) \in [r]^{s-1} \), we denote by \( R_i \) the set \( R_i = A_{i_1} \times \cdots \times A_{i_{s-1}} \times \{1\} \). According to Lemma 18, \( R_i \) is an interpolation set for every \( i \). It remains to prove that \( R_i \) and \( R_{i'} \) are disjoint under multiplication for all \( i \neq i' \). Let \( x = (x_1, \ldots, x_{s-1}, 1) \in R_i \) and \( \alpha \in \mathbb{F}_q \setminus \{0\} \). If \( \alpha \neq 1 \) then clearly \( \alpha x \notin R_{i'} \). Thus, we need to prove that \( x \notin R_{i'} \). 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 \( x \in R_i, x_j \in A_{i_j} \) and \( x_j \notin A_{i'_j} \), so \( x \notin R_{i'} \).

\(^1\)We say that \( P(x) \in \mathbb{F}_q[x_1, \ldots, x_s] \) is homogeneous if all the monomials of \( P(x) \) have the same total degree.
3.2.2 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 21.** Let $m, d, s, q$ be positive integers such that $\frac{d}{m} < q - 1$, and $C_M(m, d, s, q)$ is the multiplicity code with these parameters of length $q^s$ over the field $\mathbb{F}_q^{s(m-1)}$. Let $A \subseteq \mathbb{F}_q^s$ be an interpolation set for homogeneous polynomials of degree at most $m - 1$. Then, for every $y = (y_w)_{w \in \mathbb{F}_q^s} \in C_M(m, d, s, q)$, and for every $w_0 \in \mathbb{F}_q^s$, the set of coordinates indexed by the set

$$R = \{w_0\} + \mathbb{F}_q A \triangleq \{w_0 + \lambda v : v \in A, \lambda \in \mathbb{F}_q \setminus \{0\}\}$$

is a recovering set for the symbol $y_{w_0}$.

**Proof** The proof follows similar ideas to the one from [12]. Recall that every codeword $y = (y_w)_{w \in \mathbb{F}_q^s} \in C_M(m, d, s, q)$ corresponds to a polynomial $P(x) \in \mathbb{F}_q[x]$, of degree at most $d$, where for all $w \in \mathbb{F}_q^s$, $y_w = P^{(<m)}(w)$. We now review the recovering procedure, which will eventually prove the theorem’s statement. Every vector $v$ in the interpolation set $A$ is called a direction and will correspond to a line containing $w_0$ in the direction $v$. Reading the order $m$ evaluations of the polynomial $P(x)$ at these lines will enable us to recover the value of $P^{(<m)}(w_0)$. This procedure consists of two steps, described as follows.

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

$$p_v(\lambda) = P(w_0 + \lambda v) \in \mathbb{F}_q[x]. \quad (3.1)$$

Since the values $P(w_0 + \lambda v)$ for all $\lambda \in \mathbb{F}_q \setminus \{0\}$ are known, and $\deg(p_v) \leq d$, one can prove, as in [12], that $p_v(\lambda)$ is unique, and thus can be recovered.

For completeness, we prove that $p_v(\lambda)$ 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

$$p_1(\lambda) = p_2(\lambda) = P(w_0 + \lambda v) \text{ for all } \lambda \in \mathbb{F}_q \setminus \{0\}.$$
According to the definition of Hasse derivative, we have that

\[ P(w_0 + (\lambda_0 + \lambda)v) = p_r(\lambda_0 + \lambda) = \sum_{j=0}^{d} p_r^{(j)}(\lambda_0)(\lambda)^j, \]

and

\[ P(w_0 + (\lambda_0 + \lambda)v) = \sum_{i : wt(i) \leq d} p^{(i)}(w_0 + \lambda_0 v)(\lambda v)^i, \]

for \( r \in \{1, 2\} \). Therefore, for \( 0 \leq j \leq d \),

\[ p_r^{(j)}(\lambda_0) = \sum_{i : wt(i) = j} p^{(i)}(w_0 + \lambda_0 v)v^i. \]

Thus, given the values of \( p^{(\leq m)}(w_0 + \lambda v) \) we can calculate the values of

\[ p_r^{(\leq m)}(\lambda) = (p_r^{(0)}(\lambda), p_r^{(1)}(\lambda), \ldots, p_r^{(m-1)}(\lambda)). \]

It follows that \( p_1^{(\leq m)}(\lambda) = p_2^{(\leq m)}(\lambda) \) for every \( \lambda \in \mathbb{F}_q \setminus \{0\} \). Those two polynomials can be thought of as codewords of \( C_M(m, d, s = 1, q) \). From Lemma 15, we know that the distance of the code \( C_M(m, d, s = 1, q) \) 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, \( p_v(\lambda) \), which we now denote by

\[ p_v(\lambda) = \sum_{j=0}^{d} c_{v,j}\lambda^j. \]

**Step 2:** From (3.1), one can get that

\[ p_v(\lambda) = \sum_{i} p^{(i)}(w_0)v^i\lambda^{wt(i)} = \sum_{j=0}^{d} c_{v,j}\lambda^j, \]

and therefore we get for all \( 0 \leq j \leq d \),

\[ \sum_{i : wt(i) = j} p^{(i)}(w_0)v^i = c_{v,j}. \]

Considering only the first \( m \) of these equations, we get that \( u_i = p^{(i)}(w_0) \) is a
solution for the following set of equations

\[ \sum_{i : \text{wt}(i) = j} u_i v^i = c_{v,j}, \quad 0 \leq j < m \leq d, v \in A. \]  \hspace{1cm} (3.2)

Now we prove that the equations system (3.2) has a unique solution, and therefore the set \( R \) is a recovery set. Indeed, if we denote \( Q_j(x) = \sum_{i : \text{wt}(i) = j} u_i x^i \in \mathbb{F}_q[x_1, \ldots, x_s] \) where \( 0 \leq j < m \leq d, \) we get that the equations in (3.2) are equivalent to

\[ Q_j(v) = c_{v,j} \]

for every \( v \in A. \) 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 \( Q_j(x) \) is unique. Therefore, we can recover the value of \( P^{(<m)}(w_0) \) by solving the equations system (3.2).

**Example 3.** 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 \( P(x_1, x_2) = x_1 x_2 + x_1 x_2^2 \) inside the code \( C_M(2, 3, 2, 4). \) According to Example 2,

\[ P^{(<2)}(x_1, x_2) = (x_1 x_2 + x_1 x_2^2, x_2 + x_2^2, x_1). \]

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

\[ P^{(<2)}(1, 0) = (P^{(0,0)}(1, 0), P^{(1,0)}(1, 0), P^{(0,1)}(1, 0)). \]

First, we consider the line at direction \( v = (1, 0), \)

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

We define \( p(\lambda) = P(w + \lambda v). \) 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

\[
p(\lambda) = P(w + \lambda v) = \sum_i P^{(i)}(w) v^i \lambda^{wt(i)}.
\]

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 \( v' = (0,1) \),

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

Repeating the same steps as above for the new direction \( v' \), 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 22.** Let \( y = (y_w)_{w \in \mathbb{F}_q^s} \in C_M(m,d,s,q) \) and \( w_0 \in \mathbb{F}_q^s \). Then, \( y_{w_0} \) has \( \left\lfloor \frac{q^m}{m} \right\rfloor s^{-1} \) mutually disjoint recovering sets.

**Proof** According to Theorem 21, every interpolation set \( A \) for homogeneous polynomials of degree \( m - 1 \) defines a recovering set, which consists of the lines containing \( w_0 \) in the directions of \( v \) for all \( v \in A \). Therefore, in order to get disjoint recovering sets, all we need to do is to pick different lines. According to Lemma 20, there are \( \left\lfloor \frac{q^m}{m} \right\rfloor s^{-1} \) 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 22.

**Corollary 23.** For all \( m, d, s, q \) such that \( \frac{d}{m} < q - 1 \), the code \( C_M(m,d,s,q) \) is a \( k \)-PIR code \([q^s,n,k]^P_Q\), where \( n = \frac{(s+1)^s}{(s+1)^s - 1} \), \( k = \left\lfloor \frac{q^m}{m} \right\rfloor s^{-1} \), and \( Q = q^{(s+m-1)} \).
The next theorem summarizes the results in this section.

**Theorem 24.** For every positive integer \( s \geq 2 \), \( 0 < \alpha < 1 \), 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

\[
\begin{align*}
    k &= \Theta(n^{(1-\frac{1}{s})(1-\alpha)}), \\
    Q &= n^{\Theta(n^\alpha)}, \\
    r &= \mathcal{O}(n^{1-\frac{s}{2}}).
\end{align*}
\]

In particular, for all \( 0 \leq \epsilon < 1 \), it holds that

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

where

\[
    \delta(\epsilon) = \min_{s \geq 1} \{ \delta_s(\epsilon) \},
\]

and \( \delta_s(\epsilon) = 1 - \frac{1}{s} + \frac{\epsilon}{s-1} \). For a given value of \( \epsilon \), the value \( s^* \) that minimizes \( \delta(\epsilon) \) is \( s^* = \left\lfloor \frac{2}{1-\epsilon} \right\rfloor \).

**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 23, 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

\[
\begin{align*}
    n &= \frac{(d+s)}{(s+m-1)} = \Theta(N), \\
    k &= \Theta(q^{(s-1)(1-\alpha)}) = \Theta(N^{(1-\frac{1}{s})(1-\alpha)}) = \Theta(n^{(1-\frac{1}{s})(1-\alpha)}), \\
    Q &= q^{(s+m-1)} = q^{\Theta(n^\alpha)} = n^{\Theta(n^\alpha)}.
\end{align*}
\]

It remains to prove the claim about the redundancy. According to Lemma 15, the
redundancy of the code satisfies

\[ r = q^s - \frac{(d + s)}{(s + m - 1)} \leq q^s - \frac{d^s}{(m + s)^s} \]

\[ = \frac{(m + s)^s q^s - m^s (q - 2)^s}{(m + s)^s} \]

\[ \leq \frac{(m + s)^s q^s - m^s (q - 2)^s}{m^s} \]

\[ = O\left(\frac{q^s}{m}\right) = O(q^{s-\alpha}) \]

\[ = O(N^{1-\frac{\epsilon}{s}}) \]

\[ = O(n^{1-\frac{\epsilon}{s}}). \]

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 \( r = O(n^{1-\frac{\epsilon}{s}}) \).

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

\[ r = O(n^{1-\frac{\epsilon}{s}}) = O(n^{1-\frac{1}{s}(1-\epsilon)}) = O(n^{\delta(e)}). \]

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

\[ \frac{s^* - 2}{s^*} \leq \epsilon < \frac{s^* - 1}{s^* + 1}, \]

that is \( s^* = \left\lfloor \frac{2}{1-\epsilon} \right\rfloor \).

Lastly, in case \( n \) is not of the form \( \frac{(d + s)}{(s + m - 1)} \) we can choose the parameters \( q, m, d \) such that \( \frac{(d + s)}{(s + m - 1)} > n \), apply the construction while appending the information with zero symbols and then remove them.

In Fig. 3.1 we plot the curves of \( \delta_s(e) \) for various values of \( s \) for non-binary PIR codes.

### 3.2.3 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 $f(n) = \Omega(n^\alpha)$ if for all $\tau > 0$, $f(n) = \Omega(n^{\alpha - \tau})$. Similarly we define $f(n) = O(n^{\alpha})$ if for all $\tau > 0$, $f(n) = O(n^{\alpha + \tau})$.

**Theorem 25.** For every positive integer $2 \leq s$, $0 < \alpha < 1$, 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

$$k = \Theta\left((n / \log(n))^{(1 - \frac{1}{s})\frac{1}{\frac{1}{s} + \alpha}}\right),$$

$$r = O\left(n^{1 - \frac{\alpha}{\frac{1}{s} + \alpha}}(\log(n))^{\frac{\alpha}{\frac{1}{s} + \alpha}}\right).$$

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

$$r_p(n, k = \Omega(n^{\epsilon^-})) = O(n^{\delta(\epsilon)^+}),$$

where

$$\delta(\epsilon) = \min_{s \geq \frac{1}{\epsilon}} \{ \delta_s(\epsilon) \},$$

$$\delta_s(\epsilon) = 1 - \frac{s(1 - \epsilon) - 1}{2s(s - 1)}.$$
Proof According to Theorem 24, there exists a non-binary $k'$-PIR code $C'$ of length $N'$ over the field $\mathbb{F}_Q$ with dimension $n'$ and redundancy $r'$ such that

$$k' = \Theta((n')^{(1 - \frac{1}{s})(1 - \alpha)}),$$

$$Q = (n')^{\Theta((n')^\alpha)},$$

$$r' = \mathcal{O}((n')^{1 - \frac{2}{s}}).$$

We construct a binary PIR code $C$ from $C'$ 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 $C$ is an $[N, n, k]$-PIR code with redundancy $r = N - n$ and these parameters satisfy the following properties, where we also use that $n' = \theta(N')$,

$$n = n' \log(Q) = \Theta((n')^{1+\alpha} \log(n')),$$

$$N = N' \log(Q) = \Theta((n')^{1+\alpha} \log(n')) = \Theta(n),$$

$$k = k' = \Theta((n')^{(1 - \frac{1}{s})(1 - \alpha)}),$$

$$r = r' \log(Q) = \Theta(r'(n')^\alpha \log(n')) = \Theta((n')^{1 - \frac{2}{s} + \alpha} \log(n')).$$

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

$$k = \Theta\left((n/ \log(n))^{\left(1 - \frac{1}{s}\right)\frac{1}{1+\alpha}}\right),$$

$$r = \mathcal{O}\left(n^{1 - \frac{\alpha}{s(1+\alpha)}} (\log(n))^{\frac{\alpha}{s(1+\alpha)}}\right).$$

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

Fig. 3.2 shows the plot of $\delta_s(\epsilon)$ for various values of $s$, for binary PIR codes. The following corollary follows from the last theorem.

**Corollary 26.** For all $0 \leq \epsilon < 1$ and $n$ sufficiently large, there exists a binary $k$-PIR code $[N, n, k]_2^n$ 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 $\delta_s(\epsilon) < 1$.
Figure 3.2: Asymptotic results for binary PIR codes
The analysis so far dealt with constructing \( k \)-PIR when \( k = \Theta(n^\epsilon) \) and \( 0 \leq \epsilon < 1 \). Now we show how to use these results to construct \( k \)-PIR codes for \( \epsilon \geq 1 \). 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 [8].

**Lemma 27.** [8, Lemma 12] Let \( n, k, \) and \( k' \) be positive integers. Then, \( P(n, k + k') \leq P(n, k) + P(n, k') \).

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

**Theorem 28.** For all \( \epsilon \geq 1 \) 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 \( N = O(n^{\epsilon^+}) \).

The length achieved by the PIR construction in Theorem 28 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. 3.3 summarizes the results of binary PIR codes we achieved in this section together with the previous results. We plot the curves \( \delta_s(\epsilon) \) for \( s = 3, 5, 7, 20 \) from Theorem 25 as well as the results for \( \epsilon \geq 1 \) from Theorem 28. The lower bound on the redundancy is given by \( \Omega(\max\{k, \sqrt{m}\}) \). Therefore, if we denote \( r_p(n, k = n^\epsilon) = O(n^{\delta(\epsilon)}) \), then for \( 1 \leq \epsilon \) we get that \( \epsilon \leq \delta(\epsilon) \leq \epsilon + \rho \) for any \( \rho > 0 \).

### 3.3 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 29.** For all \( m, s, q, d, k \) such that \( d \leq m(q - km^{s-1} - 2) \) and \( k \leq \lfloor \frac{q}{m} \rfloor^{s-1} \), the code \( C_M(m, d, s, q) \) is a \( k \)-batch code \([q^s, n, k]^B_Q\), where \( n = \frac{(d+s)}{(s+1)} \) and \( Q = \frac{n}{m} \).
Theorem 28

old results

lower bound

Figure 3.3: Asymptotic results for binary PIR codes

$q^{(s+1)^{-1}}$.

**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. But since the minimum distance of $C_{M}(m,d,s = 1,q)$ equals $q - \frac{d}{m} > km^{s-1} + 1$, it can be shown in a very similar way to PIR codes, that the polynomial $p_v$ 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, thus getting the following theorem.
Theorem 30. For every $0 < \alpha < 0.5$ and $n$ sufficiently large, there exists a $k$-batch code $[N, n, k]_Q^R$ over $\mathbb{F}_Q$ of dimension $n$ such that the redundancy, $r = N - n$, the availability parameter, $k$, and the field size $Q$ satisfy

\[ k = \Theta(n^{0.5 - \alpha}), \]
\[ r = \mathcal{O}(n^{1 - \frac{\alpha}{2}}), \]
\[ Q = n^{\Theta(n^\alpha)}. \]

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

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

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

**Proof** Let $s = 2, m = \lceil q^\alpha \rceil, k = \lceil q^{1-2\alpha} \rceil$, and $d = m(q - km - 2)$. Since $km = o(q)$, we get according to Lemma 29 that $C_M(m, d, s, q)$ is a $k$-batch code $[q^2, n, k]_Q^R$, where $n = \frac{(d+2)}{m+1}$ and $Q = q^{2(m+1)}$. Since the code length is $N = q^2$, we get

\[ n = \frac{(d+2)}{(m+1)} = \Theta(N), \]
\[ k = \Theta(n^{0.5 - \alpha}), \]
\[ Q = q^{\Theta(q^\alpha)} = q^{\Theta(n^\alpha)} = n^{\Theta(n^\alpha)}. \]

The redundancy of this code is

\[ r = q^2 - \frac{(d+2)}{(m+2-1)} = q^2 - \frac{(d+1)(d+2)}{m(m+1)} \]
\[ \leq q^2 - \frac{d^2}{m(m+1)} = q^2 - \frac{m^2(q - km - 2)^2}{m(m+1)} \]
\[ = (m+1)q^2 - m(q^2 - 4q + 4 - 2(q - 2)km + k^2m^2) \]
\[ = \frac{q^2 + 2(q - 2)km^2}{m+1} + \frac{4mq - 4m - k^2m^3}{m+1} \]
\[ = \mathcal{O}(q^2/m + qkm) = \mathcal{O}(q^{2-\alpha}) = \mathcal{O}(n^{1-\frac{\alpha}{2}}). \]
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.

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

**Theorem 31.** 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

\[
\begin{align*}
k &= \Theta \left( \left( n / \log(n) \right)^{0.5 - \alpha} \right), \\
r &= \mathcal{O} \left( n^{1 - \frac{\alpha}{2}} (\log(n))^{\frac{\alpha}{2}} \right).
\end{align*}
\]

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

\[
r_B(n, k = \Theta(n^{\epsilon})) = \mathcal{O}(n^\delta(\epsilon)),
\]

where \( \delta(\epsilon) = \frac{5}{6} + \frac{\epsilon}{2} \), and \( r_B(n, k = \Theta(n^{\epsilon})) = o(n) \).

**Proof** Let \( 0 < \alpha' < 0.5 \). According to Theorem 30, there exists a \( k' \)-batch code, \( C' \), over the field \( \mathbb{F}_Q \) of length \( N' \), dimension \( n' \), redundancy \( r' \) such that

\[
\begin{align*}
k' &= \Theta \left( (n')^{0.5 - \alpha'} \right), \\
r' &= \mathcal{O} \left( (n')^{1 - \frac{\alpha'}{2}} \right), \\
Q &= (n')^{\Theta((n')^{\alpha'})}.
\end{align*}
\]

As for PIR codes, we construct binary batch code from \( C' \) 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

\[
\begin{align*}
N &= N' \log(Q) = \Theta((n')^{1 + \alpha'} \log(n')) , \\
n &= n' \log(Q) = \Theta((n')^{1 + \alpha'} \log(n')) , \\
k &= k' = \Theta((n')^{0.5 - \alpha'}) , \\
r &= r' \log(Q) = \Theta((r')^{\alpha'} \log(n')) = \Theta((n')^{1 + \frac{\alpha'}{2}} \log(n')).
\end{align*}
\]
Therefore, we get that
\[ n' = \Theta\left(\left(\frac{n}{\log(n)}\right)^{\frac{1}{1+\alpha'}}\right) \]
and
\[ k = \Theta\left(\left(\frac{n}{\log(n)}\right)^{0.5 - \frac{\alpha'}{2}}\right) = \Theta\left(\left(\frac{n}{\log(n)}\right)^{0.5 - \frac{3\alpha'}{2(1+\alpha')}}\right) \]
\[ r = O\left(n^{1-\frac{\alpha'}{2(1+\alpha')}}(\log(n))^{\frac{\alpha'}{2}}\right) \]

Denote \( \alpha = \frac{3\alpha'}{2(1+\alpha')} \). Then we get that \( 0 < \alpha \leq 0.5 \) and
\[ k = \Theta\left(\left(\frac{n}{\log(n)}\right)^{0.5 - \alpha}\right) \]
\[ r = O\left(n^{1-\frac{\alpha}{2}}(\log(n))^{\frac{\alpha}{2}}\right) \]

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

The following corollary follows from the last theorem.

**Corollary 32.** For all \( 0 \leq \epsilon < 0.5 \) 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 \( \epsilon \geq 0.5 \) and construct binary batch codes \([N, n, k]_B\) such that \( k = \Theta(n^\epsilon) \) and \( N = O(n^{0.5+\epsilon}) \), for \( \epsilon \geq 0.5 \). However, in section 4.4 we show another method to construct batch codes which achieve better results in the case \( \epsilon \geq 0.5 \). In particular, we show that \( r_B(n,k = n^\epsilon) = o(n) \) when \( 0 < \epsilon < 1 \), and \( r_B(n,k = n^\epsilon) = O(n^{\epsilon^2}) \) when \( 1 \leq \epsilon \).
Chapter 4

The Array Construction

4.1 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 [10] which was also used in [8] 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 \(x\) of length \(n = s^2\), it is represented as a two-dimensional array of size \(s \times s\) as follows

\[
\begin{array}{cccc}
    x_0 & x_1 & \ldots & x_{s-1} \\
    x_s & x_{s+1} & \ldots & x_{2s-1} \\
    \vdots & \vdots & \ddots & \vdots \\
    x_{s^2-s} & x_{s^2-s+1} & \ldots & x_{s^2-1}
\end{array}
\]

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

\[
\begin{array}{cccc}
    x_0 & x_1 & \ldots & x_{s-1} & r_0 \\
    x_s & x_{s+1} & \ldots & x_{2s-1} & r_1 \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    x_{s^2-s} & x_{s^2-s+1} & \ldots & x_{s^2-1} & r_{s-1} \\
    c_0 & c_1 & \ldots & c_{s-1}
\end{array}
\]
where $r_i, c_i$ is the sum of the bits in the $i$th 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 [3] 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 $(x \mod m)$.

**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 $t \in [p]$,

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

(4.1)

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

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

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

Lastly, the code \( C_A(r, p, S) \) is defined to be

\[ C_A(r, p, S) = \{(x, E_{r,p,S}(x)) : x \in \{0, 1\}^n\}. \]

The following example demonstrates the Array Construction.

**Example 4.** Let \( r = 3, p = 5, n = 15 \), and assume we want \( k = 4 \) recovering sets. First, we arrange the input, \( x \), in a \( 3 \times 5 \) table

\[
\begin{array}{cccc}
  x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} & x_{0,4} \\
  x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\
  x_{2,0} & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\
\end{array}
\]

We consider diagonals corresponding for slopes 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

\[
\begin{align*}
\rho_{0,0} &= x_{0,0} + x_{1,0} + x_{2,0} \\
\rho_{0,1} &= x_{0,1} + x_{1,1} + x_{2,1} \\
\rho_{0,2} &= x_{0,2} + x_{1,2} + x_{2,2} \\
\rho_{0,3} &= x_{0,3} + x_{1,3} + x_{2,3} \\
\rho_{0,4} &= x_{0,4} + x_{1,4} + x_{2,4} \\
\end{align*}
\]

For \( s = 1 \), the following table shows the corresponding diagonals.

\[
\begin{array}{cccc}
  x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} & x_{0,4} \\
  x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\
  x_{2,0} & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\
\end{array}
\]

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.

<table>
<thead>
<tr>
<th></th>
<th>( x_{0,0} )</th>
<th>( x_{0,1} )</th>
<th>( x_{0,2} )</th>
<th>( x_{0,3} )</th>
<th>( x_{0,4} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_{1,0} )</td>
<td>( x_{1,1} )</td>
<td>( x_{1,2} )</td>
<td>( x_{1,3} )</td>
<td>( x_{1,4} )</td>
<td></td>
</tr>
<tr>
<td>( x_{2,0} )</td>
<td>( x_{2,1} )</td>
<td>( x_{2,2} )</td>
<td>( x_{2,3} )</td>
<td>( x_{2,4} )</td>
<td></td>
</tr>
</tbody>
</table>

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, \( x \) is encoded to

<table>
<thead>
<tr>
<th></th>
<th>( x_{0,0} )</th>
<th>( x_{0,1} )</th>
<th>( x_{0,2} )</th>
<th>( x_{0,3} )</th>
<th>( x_{0,4} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_{1,0} )</td>
<td>( x_{1,1} )</td>
<td>( x_{1,2} )</td>
<td>( x_{1,3} )</td>
<td>( x_{1,4} )</td>
<td></td>
</tr>
<tr>
<td>( x_{2,0} )</td>
<td>( x_{2,1} )</td>
<td>( x_{2,2} )</td>
<td>( x_{2,3} )</td>
<td>( x_{2,4} )</td>
<td></td>
</tr>
<tr>
<td>( \rho_{0,0} )</td>
<td>( \rho_{0,1} )</td>
<td>( \rho_{0,2} )</td>
<td>( \rho_{0,3} )</td>
<td>( \rho_{0,4} )</td>
<td></td>
</tr>
<tr>
<td>( \rho_{1,0} )</td>
<td>( \rho_{1,1} )</td>
<td>( \rho_{1,2} )</td>
<td>( \rho_{1,3} )</td>
<td>( \rho_{1,4} )</td>
<td></td>
</tr>
<tr>
<td>( \rho_{2,0} )</td>
<td>( \rho_{2,1} )</td>
<td>( \rho_{2,2} )</td>
<td>( \rho_{2,3} )</td>
<td>( \rho_{2,4} )</td>
<td></td>
</tr>
</tbody>
</table>

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 prove several useful properties.

**Lemma 35.** For all \( r, p, \) and \( s \in [p] \) the set \( P_s(r, p) \) is a partition of \( [r] \times [p] \).

**Proof** 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 $0 \leq t_1 < t_2 \leq p - 1$ such that $D_{s,t_1} \cap D_{s,t_2} \neq \emptyset$. Hence there exist $0 \leq i_1, i_2 \leq r - 1$ 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.

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

**Lemma 36.** For all $r \leq p$ 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 $t_1, t_2 \in [p]$, $|D_{s_1,t_1} \cap D_{s_2,t_2}| \leq 1$.

**Proof** 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

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

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

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

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.

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

37
Theorem 37. For all $r \leq p$ and $S \subseteq [p]$, such that $|S| = k - 1 \leq p$. 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 $C_A(r, p, S)$ 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 $x = (x_{i,j})_{(i,j) \in [r] \times [p]}$ and $E_{rp,p,S}(x) = (\rho_{t,t'})_{(t,t') \in [k-1] \times [p]}$, 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 35, 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 36, these sets intersect only on the entry $(i, j)$. Therefore, we conclude that the following $k - 1$ sets

$$R_\ell = \{\rho_{t,t'}\} \cup \{x_{i,j'} : (i', j') \in D_{s_{\ell},t_{\ell}} \setminus \{(i, j)\}\},$$

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, 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 38. Let $n = p^2$, where $p$ is a prime number, and $k \leq \sqrt{n}$. The code $C_A(r = p, p, S = [k - 1])$ is a $k$-PIR code with redundancy $(k - 1)\sqrt{n}$. In particular, for all $k \leq \sqrt{n}$, $r_p(n, k) = O(k\sqrt{n})$.

It can be shown using the DTI codes from [14] that the result $r_p(n, k) = O(k\sqrt{n})$ 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.

38
**Remark 1.** It shall be noted that the partition $P_s(p, p)$ generates a latin square (see [11], 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 [11], Chapter 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.

### 4.2 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 with constructions of LRCs, it does not necessarily result with better codes than existing ones. However, this construction can result with good batch codes as well as batch codes with restricted size for the recovering sets [30]. 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.

#### 4.2.1 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 [20] and is stated below.

**Lemma 39.** [20] Let $C$ 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 $i$th bit intersects with at most one recovering set of the $j$th bit. Then, the code $C$ is an $(r, k)$-batch code.

From the above lemma, it is readily verified that the code $C_A(2, n/2, S = [k])$ 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 39. For that, we use the following definition.
\textbf{Definition 40.} 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 \textbf{\( r \)-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
\[
x s_1 + y s_2 = (x + y) s_3 \tag{4.2}
\]
We say that the set \( S \) does not contain an \textbf{\( r \)-weighted arithmetic progression modulo} \( p \) if equation (4.2) does not hold modulo \( p \).

Given this definition, we prove the following theorem.

\textbf{Theorem 41.} Let \( C_A(r, p, S) \) be a code from the Array Construction, where \( r \leq p \) and \( S \subseteq \{p\}, |S| = k \leq p \). If one of the following two conditions holds

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

then the code \( C_A(r, p, S) \) is an \((r, k)\)-batch code of dimension \( rp \) and redundancy \( kp \).

\textbf{Proof} Assume that the set \( S = \{s_0, s_1, \ldots, s_{k-1}\} \) is in an increasing order. According to Theorem 37, the code \( C_A(r, p, S) \) is an \((r, k + 1)\)-PIR code. Therefore, it remains to prove that the code \( C_A(r, p, S) \) satisfies the condition of Lemma 39 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 37. In particular, for \( \ell \in [k] \), the \( \ell \)th recovering set of \((i, j) \) is
\[
R^{(i,j)}_{\ell} = \{r_{\ell,t}\} \cup \{x_{i',j'} : (i', j') \in D_{s_\ell,t}\} \setminus \{(i, j)\},
\]
for some \( t_{\ell} \in [p] \), and we denote \( D(R^{(i,j)}_{\ell}) = D_{s_\ell,t_{\ell}} \). 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 \( R^{(i,j)}_{\ell_1} \) that intersects with two recovering sets \( R^{(i',j')}_{\ell_1}, R^{(i',j')}_{\ell_2} \) of \((i', j') \). Assume that \( b_1 \in R^{(i,j)}_{\ell_1} \cap R^{(i',j')}_{\ell_1} \) and \( b_2 \in R^{(i,j)}_{\ell_1} \cap R^{(i',j')}_{\ell_2} \) where \( b_1, b_2 \) are
codeword entries. Assume that

\[ D(R^{(i,j)}_{i_1}) = D_{s_1',t_1} \]
\[ D(R^{(i',j')}_{i_1'}) = D_{s_2',t_2} \]
\[ D(R^{(i',j')}_{i_2'}) = D_{s_3',t_3} \]

for some \( s_1', s_2', s_3' \in S \) and \( t_1, t_2, t_3 \in [p] \). Since \( (i', j') \in D_{s_1', t_1} \cap D_{s_2', t_2} \), we deduce by Lemma 36 that \( D_{s_1', t_1} \cap D_{s_2', t_2} \cap D_{s_3', t_3} = \emptyset \). 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 \( (D_{s_2', t_2} \setminus \{(i, j)\}) \cap (D_{s_3', t_3} \setminus \{(i', j')\}) = \emptyset \). 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

\[ (i_1, j_1), (i_2, j_2) \in D_{s_1', t_1}, \]
\[ (i_1, j_1), (i', j') \in D_{s_2', t_2'}, \]
\[ (i_2, j_2), (i', j') \in D_{s_3', t_3}. \]

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:

\[ j_1 = (t_1 + i_1 s_1')_p, \quad j_2 = (t_1 + i_2 s_1')_p, \]
\[ j_1 = (t_2 + i_2 s_2')_p, \quad j' = (t_2 + i' s_2')_p, \]
\[ j_2 = (t_3 + i_2 s_3')_p, \quad j' = (t_3 + i' s_3')_p. \]

This implies that

\[ \langle j_2 - j_1 \rangle_p = \langle (i_2 - i_1) s_1' \rangle_p, \]
\[ \langle j' - j_1 \rangle_p = \langle (i' - i_1) s_2' \rangle_p, \]
\[ \langle j' - j_2 \rangle_p = \langle (i' - i_2) s_3' \rangle_p. \]

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

$$\langle x s'_1 + y s'_2 \rangle_p = \langle (x + y) s'_3 \rangle_p, \quad (4.3)$$

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 $xs'_1 + ys'_2 \leq \max_{s \in S} \{s\} \cdot (r - 1) < p$, 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 41. 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 $k \leq \frac{\log(p/r)}{\ell} + 1$ 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 41 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.

4.2.2 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 41 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 $S = \{s_0, s_1, \ldots, s_{k-1}\}$.

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

**Theorem 42.** 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 $p > 2k^2r^2$. 


Figure 4.1: The Greedy Algorithm

**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

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

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 43.** 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 43 we are now at a point to construct $k$-batch codes with good redundancy when $k$ is a function of $n$.

**Corollary 44.** For any $n$ and $k$ such that $k = o(\sqrt{n})$, there exists a $k$-batch code of dimension $n$ and redundancy $O(n^{3}k^{5})$. In particular, for $0 < \epsilon < 1/2$, it holds that

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

**Proof** For $n$ and $k$, let us choose $r = \lceil n^{1/3}k^{2} \rceil$, and $p$ is the smallest prime number such that $2k^2r^2 < p$. Then, according to Theorem 43, there exists an $(r, k)$-batch code of dimension $pr > n$ and redundancy $kp$. That is, the redundancy satisfies

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

The second statement in the corollary is established for $k = n^\epsilon$ in the last equation.
4.2.3 Construction of \((r,k)\)-Batch Codes

In this part we construct sets that satisfy the second condition in Theorem 41 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. In [7], Erdős and Turan initiated the study of sequences that do not contain three terms in arithmetical progression\(^1\), 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. The following theorem was proved by Behrend in [2].

**Theorem 45.** 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 41 and Theorem 45.

**Corollary 46.** 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 47** For any integers \(r, d \geq 2\), \(n \geq 2\), \(k \leq n(d-1)^2\), we define the set

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

**Lemma 48.** 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

\(^1\)in fact, they called them A sequences but subsequent papers used the term without 3-term arithmetic progression.
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 $0 \leq a_i < rd - 1$ for $1 \leq i \leq n$. We define $\text{norm}(A)$ to be the Euclidean norm of $u(A)$, i.e., $\text{norm}(A) = ||u(A)|| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$. Then,

\[
\text{norm}(xA_1 + yA_2) = \text{norm}((x + y)A_3) = (x + y)\sqrt{k},
\]

\[
\text{norm}(xA_1) + \text{norm}(yA_2) = (x + y)\sqrt{k}.
\]

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

\[
\text{norm}(xA_1 + yA_2) = ||u(xA_1 + yA_2)||
\]

\[
= ||xu(A_1) + yu(A_2)||
\]

\[
\leq ||xu(A_1)|| + ||yu(A_2)||
\]

\[
= \text{norm}(xA_1) + \text{norm}(yA_2).
\]

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 49.** Let $r$ be a positive integer and $0 < \alpha < 1$. Then for all $\varepsilon > 0$ and sufficiently large $p$, $v_r(p) > p^{\frac{1-3log_2\alpha}{\sqrt{log_2 p}}}$. In particular, for any fixed $r$ there exists a set $R \subseteq [p]$ that does not contain an $r$-weighted arithmetic progression such that $|R| = \Omega(p^n)$.

**Proof** There are $d^n$ different vectors $(a_1, a_2, \ldots, a_n)$ that satisfy $0 \leq a_i < d$, 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

\[
v_r((rd - 1)^n) \geq \frac{d^n}{n(d - 1)^2 + 1} > \frac{d^{n-2}}{n}.
\]
Let $p$ be given. Choose $n = \left\lfloor \sqrt{\frac{\log(p)}{\log(r)}} \right\rfloor$, and $d$ such that
\[(rd - 1)^n \leq p < (rd + r - 1)^n.\]

Then,
\[v_r(p) \geq v_r((rd - 1)^n) > \frac{d^{n-2}}{n} > \left(\frac{p^{1/n} - (r - 1)^{n-2}}{n^r} - 1\right)\frac{(1 - (r - 1)p^{-1/n})^{n-2}}{n^r}.
\]

Since $(1 - (r - 1)p^{-1/n})^{n-2} \xrightarrow{p \to \infty} 1$, it follows that for sufficiently large $p$
\[v_r(p) > \frac{p^{1-2/n}}{n^r} = p^{1-\frac{2}{n}} - \frac{\log n}{\log p} - \frac{(n-1)\log r}{\log p} > p^{1-\frac{3\log r + \epsilon}{\sqrt{\log p}}},\]
for any $\epsilon > 0$.

Thus, we get the following result.

**Theorem 50.** For every $0 < \alpha < 1$, $k = \mathcal{O}(n^\alpha)$, and fixed $r \geq 3$, there exists an $(r, k)$-batch code of dimension $n$ and rate $\frac{r}{r+k}$.

### 4.3 Improved Construction of Batch Codes from the Array Construction

In this section, we provide a generalization for the array construction, which can result 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 diagonals in the array and the resulting code will still be a batch code.

**Construction 51 (Generalized Array Construction)** Let $\mathbb{F}$ be a finite field of size $q$, and let $r, p, n, k$ be positive integers such that $k \leq p$, $n = rp$, and $S \subseteq [p]$ be...
a subset of size $k$. We denote $S = \{s_0, s_1, \ldots, s_{k-1}\}$ where $0 \leq s_0 < s_1 < \cdots < s_{k-1} \leq p - 1$. Let $\varepsilon : \mathbb{F}^r \rightarrow \mathbb{F}^{r+m}$ 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 $E_{r, p, S, \varepsilon} : \mathbb{F}^m \rightarrow \mathbb{F}^{k \cdot p \cdot m}$ as follows. The input vector $x \in \mathbb{F}^m$ is represented as an $r \times p$ array, that is $x = (x_{i,j})_{(i,j)\in[r]\times[p]}$ 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 $t \in [p]$,

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

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

$$
C_G(r, p, S, \varepsilon) = \{(x, E_{r, p, S, \varepsilon}(x)) : x \in \mathbb{F}^m\}.
$$

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 52.** Let $\mathbb{F}$ be a finite field, $r, p, k$ be positive integers and $S \subseteq [p]$ be a set of size $k - 1 \leq p$. Let $\varepsilon$ 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 $C = C_G(r, p, S, \varepsilon)$ 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 multisets 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 35, for every $\ell \in [w]$, the partition set $P_{s_{\ell}}(r, p)$ has a set that contains the entry $(i_{\ell}, j_{\ell})$. 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 $I_{\ell}$ denote the set of all the symbols that are contained in $D_{s_{\ell}, t_{\ell}}$, and
and another set $D_{s_\ell',t_\ell'}$ for $\ell' \neq \ell \in [w]$ or $U$. Formally,

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

According to Lemma 36, $|D_{s_\ell_1,t_\ell_1} \cap D_{s_\ell_2,t_\ell_2}| \leq 1$ for any $\ell_1 \neq \ell_2 \in [k-1]$, and $|U| = k - w$, therefore it follows that $|I_\ell| \leq |U| + w - 1 = k - 1$ for any $\ell \in [w]$. Now, for $\ell \in [w]$, we define

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

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 $I_\ell$ from every set. It remains to prove that $R_\ell$ is a recovery set for $(i_\ell,j_\ell)$ when $\ell \in [w]$. To this end, notice that the vector $c = (v, \rho_{\ell,t_\ell})$, where $v = (x_{i,j} : (i,j) \in D_{s_\ell,t_\ell})$, is a codeword of a code with minimum distance $k$. Reading the symbols of $R_\ell$ means we are able to construct the vector $c_e = (v_e, \rho_{\ell,t_\ell})$, where $v_e = (x_{i,j} : (i,j) \in R_\ell)$. Since $|I_\ell| \leq k - 1$, it follows that $c_e$ can be received from $c$ by erasing at most $k - 1$ entries, and therefore $c$ can be recovered from $c_e$. Finally, since $(i_\ell,j_\ell) \in D_{s_\ell,t_\ell}$, we have also recovered the symbol at position $(i_\ell,j_\ell)$.

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

**Theorem 53.** Let $n$, $p$, $q$ be positive integers such that $n = p^2$, $p$ is a prime number and $p + k - 1 < q$. Then for $k \leq p + 1$, 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 52 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 54.** 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 55.** 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 $O((k-1)^2\sqrt{n}\log(n))$.

**Proof** According to Lemma 52, it suffices to find a binary code $C$ of dimension $p$, minimum distance $k$, and redundancy $O((k-1)\log(p))$. To accomplish this task for $k < p/\log p$, we use BCH codes (see [21], Chapter 8.4), and therefore the theorem is proved.

The following corollary follows directly from the last theorem.

**Corollary 56.** For $k \leq \sqrt{n}/\log n$, it holds that $$r_B(n,k) = O((k-1)^2\sqrt{n}\log(n)).$$ In particular, for $0 < \epsilon \leq 0.25$, it holds that $$r_B(n,k = n^\epsilon) = O(n^{0.5+2\epsilon}),$$ and $r_B(n,k) = O(\sqrt{n}\log(n))$ for fixed $k$.

It should be noted that [23] constructed binary $k$-batch codes with redundancy $O(\sqrt{n}\log(n))$ 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 $r_B(n,k) = O(k^2\sqrt{n}\log(n))$ while [23] get that $r_B(n,k) = O(k^{2\epsilon/k^2}\sqrt{n}\log(n))$ for fixed $k$.

### 4.4 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 3.2, 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.

**Lemma 57.** Let $C$ be an $[N,n,k]^P$ PIR code. Then, $C$ is also an $(r = \frac{2N}{k}, k' = \frac{k}{2})$-PIR code of dimension $n$ and length $N$. 

49
Proof We prove that for every bit there exist \( k' = k/2 \) disjoint recovery sets of size at most \( r = \frac{2N}{k} \). Since \( C \) 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 58. Let \( C \) be an \((r,k)\)-PIR code of dimension \( n \) and length \( N \). Then, \( C \) is an \([N,n,k']^B\) batch code, where \( k' = \lfloor k/r \rfloor \).

Proof Let \( R = \{i_1,i_2,\ldots,i_r\} \) be the multiset of requested bits. Since \( C \) is an \((r,k)\)-PIR code, we get that for every \( j \in [k] \) there exist \( k \) disjoint recovery sets \( S_{i_1}, S_{i_2}, \ldots, S_{i_k} \) 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 \( S_{i_1} \). 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 \( i_\ell \). The number of bits contained in the chosen sets is at most \((\ell - 1)r < k\), therefore there exists a recovery set of \( i_\ell \) which does not intersect with the chosen sets. We choose this set to be the recovery set of \( i_\ell \).

From the last two lemmas we conclude the following corollary.

Corollary 59. Let \( C \) be an \([N,n,k]^P\) PIR code. Then, \( C \) 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 60. For \( 0 < \epsilon < 1 \), it holds that

\[
r_B(n,n^\epsilon) \leq r_p(n,n^{\frac{\epsilon+1}{2}}).
\]

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 \( N \leq 2n \). Thus, by applying Corollary 59 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 25 from section 3.2.
Corollary 61. For $0 < \epsilon < 1$, it holds that

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

where

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

$$\delta_s(\epsilon) = 1 - \frac{s(1 - \epsilon) - 1}{2s(s - 1)}.$$

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

**Theorem 62.** For all $\epsilon \geq 1$ and $n$ sufficiently large, there exists a binary $k$-batch code $[N, n, k]_B^s$ of dimension $n$ such that $k = \Theta(n^\epsilon)$ and $N = \mathcal{O}(n^{\epsilon^+}).$

Let us denote

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

In Fig. 4.2 we plot the results on the asymptotic behavior of the redundancy of batch codes. These plots are received from Theorem 31 from Chapter 3.3, and Corollary 44 from Chapter 4.2, and Corollary 56 from Chapter 4.3, and Corollary 61 and Theorem 62 from Chapter 4.4.

### 4.5 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 [23]. 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.

**Theorem 63.** Let $n = p^2$ where $p$ is a prime number. The code $C$, that extends $C_A(r = p, p, S = \{4\})$ by adding one all-parity bit, is a 5-batch code with redundancy $4p + 1 = \theta(\sqrt{n})$. 

51
Proof Let $x = (x_{i,j})_{(i,j)\in[p]\times[p]}$ be the information bits, which are systematically encoded to $y = (x, \rho_1, \ldots, \rho_{4p+1})$. We denote the requested bits by $R = \{(i_0, j_0), (i_1, j_1), \ldots, (i_4, j_4)\}$. According to Theorem 37, the code $C$ 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 [23].

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 64.

Case 3: $R = \{(i_0, j_0), (i_0, j_0), (i_1, j_1), (i_1, j_1), (i_1, j_1)\}$. Since $C$ 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 $y$. 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 $\sum_{i=1}^{4} \sum_{x \in S_i} x = 0$. 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.

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 64.** Let $n = p^2$ where $p$ is prime, and let $5 \leq k$ be a positive integer. Assume that $\frac{(k-2)(k-3)}{2} < p$. Then, the code $C = C_A(r = p, p, S = \lfloor \frac{(k-2)(k-3)}{2} + 1 \rfloor)$ is a $k$-batch code under the constraint that each bit is requested at most twice.

**Proof** Let $x = (x_{i,j})_{(i,j)\in[p]\times[p]}$ be the information symbols. Since for $5 \leq k$ it holds that $\frac{(k-2)(k-3)}{2} + 2 \geq k$, the code $C$ is a $k$-PIR code according to Theorem 37, and therefore if at most one bit is requested twice then this can be easily solved according to Lemma 3 from [23]. Otherwise, let $R = \{(i_0, j_0), (i_1, j_1), \ldots, (i_{r-1}, j_{r-1})\}$ be the requested bits, $r \leq k - 2$, where some of these bits are requested twice. According to Construction 34 and the proof of Theorem 37, 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 36, 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 35, these sets do not intersect. Therefore, we conclude that the following $2r$ sets

$$R_1 = \{x_{i,j} : (i, j) \in D_{0,t_1} \setminus \{(i, j)\}\}$$

$$R_2 = \{x_{i,j} : (i, j) \in D_{0,t_2} \setminus \{(i, j)\}\}$$

for $\ell \in [r]$, are mutually disjoint recovering sets for the requested bits.
Chapter 5

Conclusions and Future Work

5.1 Summary

In this thesis, we presented new construction of PIR and batch codes which attempt to close the gap between the lower and upper bound of the redundancies of these codes. Our results are based on two main constructions; Multiplicity codes which generalize Reed Muller codes and the Array Construction.

In particular, 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. We list some of the results in this work:

1. For $k = o(n)$, we showed that there exist asymptotically optimal binary $k$-PIR and $k$-batch codes with rate approaching one. In particular, for $0 < \epsilon < 1$, it holds that

$$r_P(n, k = n^\epsilon) \leq r_B(n, k = n^\epsilon) = o(n).$$

Fig. 3.3 summarizes the results for the PIR case when $k = n^\epsilon$, while Fig. 4.2 shows the results for batch codes.

2. We constructed nearly optimal PIR and batch codes for $k \geq n$. More specifically, we showed that for $\epsilon \geq 1$, it holds that

$$r_P(n, k = n^\epsilon) \leq r_B(n, k = n^\epsilon) = O(n^{\epsilon+}).$$

3. Construction of asymptotically optimal non-binary batch codes for fixed $k$. 

54
In other words, we constructed $k$-batch codes of dimension $n$ with redundancy $\mathcal{O}((k - 1)^2 \sqrt{n})$, when the field size is at least $\sqrt{n} + k - 1$.

4. For $k = 5$, we proved that

$$r_B(n,k = 5) = \Theta(\sqrt{n}).$$

5. For fixed $k \geq 6$, we proved that

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

6. We presented a method for constructing batch codes from PIR codes. In particular, this method implies that

$$r_B(n,k = n^\epsilon) \leq r_P(n,k = n^{(\epsilon + 1)/2}).$$

### 5.2 Future Work

While this work improved the upper bound on the redundancy of PIR and batch codes, there are still many open questions to be solved. In general, any progress in tightening any gap between the lower and upper bounds for different values of $k$ would be interesting. We list the following directions for future work:

1. Constructions of binary batch codes that achieve optimal redundancy for fixed value of $k$. This work constructed optimal non-binary batch codes for fixed $k$, and optimal binary batch codes for $k = 5$. However, for $k > 6$, it is only known that $r_B(n,k) = \mathcal{O}(\sqrt{n} \log n)$. Our conjecture is that for any fixed $k$, $r_B(n,k) = \Theta(\sqrt{n})$.

2. Introducing new constructions which improve the current upper bound for PIR and batch codes when $k = n^\epsilon$. In particular, we strongly believe that $r_P(n,k) = \Theta(\sqrt{kn})$ for $k < n$.

3. It is interesting to see that the Generalized Array Construction is used in this work mainly in two forms to construct batch codes. One is when the internal encoder has distance 2 (called as the Array Construction), where we need to choose the diagonals carefully in order to satisfy the requirements of batch codes, and another form when the internal encoder has distance $k$, where we
do not have any requirement over the set of chosen diagonals. An interesting direction would be to understand what happens when the distance is between 2 and \( k \), and check whether this can improve the upper bound.

4. The Array Construction which was described in this paper is defined only as a two dimensional array. It would be useful to generalize this construction to larger dimensions, and check the improvement on the redundancy.

5. Even though this work does not introduce new results for the lower bound of the codes studied in this thesis, 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.
Bibliography


בניית קוד יניון لنוש לחסוך מב keer

הלם עסיפי
ב니ית קודי נגישה לאחסון מבוצר

חיבור על מחקר

לשם מייליו חלקי של הדירישות לקבנול התנזור
מגיסטר למעידים ובמורי המוחשב

הלאל עאסי

名家 להנחת ההכנתו – מכון טכנולוגי לישראל
חיפה תשע"ז ~ ספטמבר 2017
המחק נעשה בהנחיית פרופ' איאן יעקובי בפקולטה למדעי המחשב.

תודה

אני מודה להנחת של פרופ' איאן יעקובי על התמיכת וה器材воротה לאורכך כלたり.

ברצוני גם להודות להברית של(eventName) על חופש זמן חברי התמדורות בטכנון.

בسكو, ברצוני להודות להרימ של(EventName) על העידון והתמודיכות הבולטים מתניה.

אני מודה לטכניזוק על התמיכת הבספית והתופישה במחלקות.
In today's era, with the advent of the internet and social networks, there has been a significant increase in the amount of information produced worldwide. One of the main factors that increase the rate of news production is the content of users. Websites, bloggers, forums, and social networks have brought it to the point where every person in the world can create a lot of information in a single day. The main challenge now lies in the hands of researchers who need to design and implement storage management systems that can cope with the explosion of information. These storage systems need to be organized into a large number of different storage devices, known as a cluster, in a way that they can work together to overcome the challenges of each part of the system, primarily due to failures and partial failures of some of the storage devices. The simplest and most widespread way to do this is to replicate the data a certain number of times (in this case, three times). For this, there are many advantages in terms of ease of implementation, speed of recovery from failures, and easy access to the information. On the other hand, to solve this problem, a large memory is required for the storage systems of companies such as Google, Microsoft, and Facebook.

In addition to data recovery and accessibility, solving this problem is also important for improving the performance of these storage systems. This feature could be useful in storage systems with a large number of users. The main reason for this is that each file has an exact time (or a possibility to do so), and if we want to retrieve a file, the system must deal with the requests in a sequential manner, which leads to significant delays in the time response of the system.

This dissertation deals with a new type of codes known as one-step majority-logic decodable codes. The first major work in this area was done by Ashikhmin in the late 1990s, who defined them as codes with a single majority decoder. Despite the fact that a single majority decoder is simple and efficient, it is not very flexible, and it may not be able to meet the needs of a storage system with a large number of different users. On the other hand, a storage system with a large number of users may require a more flexible and efficient decoder, especially in terms of handling partial failures. This is where the concept of a one-step majority-logic decodable code comes in. These codes are designed to overcome the limitations of a single majority decoder while maintaining the simplicity and efficiency of the decoding process.
The main objective is to build a code with minimal modification. More formally, given \( N \), the goal is to find the maximum number of symbols \( n \) with minimal modification \( n \). For \( B(n,k) = 1 \), it is known that \( k = 5 \) for the family of private information retrieval (PIR) codes. For \( B(n,k) = 2 \), it is known that \( k = 3 \) or \( k = 4 \). Therefore, we focus on the case \( k \geq 5 \) for the minimum modification of 2. Such codes are called \( k \)-optimal codes and are of interest in the context of this thesis. The general bounds for \( B(n,k) \) are determined.

The function \( \Theta(\sqrt{n}) \) is asymptotically tight as \( n \to \infty \).

Consider a function \( f(n) \) that satisfies \( f(n) = O(\sqrt{n}) \). Then, \( f(n) = \Omega(\sqrt{n}) \) as \( n \to \infty \).

For any constant \( k > 5 \), we have

\[
\mathbb{P}(\epsilon) = \Omega(n^{\frac{1}{2}}) \quad \text{for} \quad \epsilon = \frac{1}{\sqrt{n}}.
\]

In particular, this bound holds for \( k = 3 \).}

The construction of the codes is based on the fact that the number of codewords is \( O(\sqrt{n}) \).}

For any \( k > 5 \), we have

\[
r_B(n, k) = O(k^2 \sqrt{n} \log(n)).
\]

In particular, this bound holds for \( k = 3 \).}

The construction of the codes is based on the fact that the number of codewords is \( O(\sqrt{n}) \).}

For any \( k > 5 \), we have

\[
r_B(n, k) = O(k^2 \sqrt{n} \log(n)).
\]

In particular, this bound holds for \( k = 3 \).