CODING FOR THE CLOUDS: CODING TECHNIQUES FOR ENABLING SECURITY, LOCALITY, AND AVAILABILITY IN DISTRIBUTED STORAGE SYSTEMS

A Dissertation

by

SWANAND RAVINDRA KADHE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Alex Sprintson
Committee Members,	P. R. Kumar
	Krishna Narayanan
	J. Maurice Rojas
Head of Department,	Miroslav M. Begovic

December 2017

Major Subject: Electrical Engineering

ABSTRACT

Cloud systems have become the backbone of many applications such as multimedia streaming, e-commerce, and cluster computing. At the foundation of any cloud architecture lies a large-scale, distributed, data storage system. To accommodate the massive amount of data being stored on the cloud, these distributed storage systems (DSS) have been scaled to contain hundreds to thousands of nodes that are connected through a networking infrastructure. Such data-centers are usually built out of commodity components, which make failures the norm rather than the exception.

In order to combat node failures, data is typically stored in a redundant fashion. Due to the exponential data growth rate, many DSS are beginning to resort to error control coding over conventional replication methods, as coding offers high storage space efficiency. This paradigm shift from replication to coding, along with the need to guarantee reliability, efficiency, and security in DSS, has created a new set of challenges and opportunities, opening up a new area of research. This thesis addresses several of these challenges and opportunities by broadly making the following contributions. (i) We design practically amenable, low-complexity coding schemes that guarantee security of cloud systems, ensure quick recovery from failures, and provide high availability for retrieving partial information; and (ii) We analyze fundamental performance limits and optimal trade-offs between the key performance metrics of these coding schemes.

More specifically, we first consider the problem of achieving information-theoretic security in DSS against an eavesdropper that can observe a limited number of nodes. We present a framework that enables design of secure *repair-efficient* codes through a joint construction of inner and outer codes. Then, we consider a practically appealing notion of *weakly secure coding*, and construct coset codes that can weakly secure a wide class of *regenerating codes* that reduce the amount of data downloaded during node repair.

Second, we consider the problem of meeting *repair locality* constraints, which specify the number of nodes participating in the repair process. We propose a notion of *unequal locality*, which enables different locality values for different nodes, ensuring quick recovery for nodes storing important data. We establish tight upper bounds on the minimum distance of linear codes with unequal locality, and present optimal code constructions. Next, we extend the notion of locality from the Hamming metric to the *rank* and *subspace* metrics, with the goal of designing codes for efficient data recovery from special types of correlated failures in DSS. We construct a family of locally recoverable rank-metric codes with optimal data recovery properties.

Finally, we consider the problem of providing high *availability*, which is ensured by enabling node repair from multiple disjoint subsets of nodes of small size. We study codes with availability from a queuing-theoretical perspective by analyzing the average time necessary to download a block of data under the Poisson request arrival model when each node takes a random amount of time to fetch its contents. We compare the delay performance of the availability codes with several alternatives such as conventional erasure codes and replication schemes.

DEDICATION

To my parents, Aai and Baba, for their unconditional love, support, and encouragement.

ACKNOWLEDGMENTS

As this journey of five and a half years comes to an end, it is time to pause for a moment to thank the many people who have contributed in making the journey a wonderful, rewarding, and memorable experience.

First and foremost, I would like to express my deepest gratitude to my advisor Prof. Alex Sprintson for his incredible support and encouragement. He not only gave me the freedom to pursue new research endeavors, but also he was always there whenever I needed his help. He has always been available for discussion any time of day and night (literally!), whether it is for brainstorming new ideas, or solving issues while working against a deadline, or for talking about current affairs. His ingenuity to convert a complex research problem into a fun puzzle and his out-of-the-box thinking has always amazed me. He has always encouraged and helped me to build new collaborations, which have played an instrumental role in my growth as a researcher. Alex, I thank you with all my heart for being so caring, compassionate, and supportive!

I would like to express my sincere thanks to Prof. P. R. Kumar, Prof. Krishna Narayanan, and Prof. J. Maurice Rojas for serving on my dissertation committee and providing critical feedback. Maurice often presented a different viewpoint on the research problems, stemming from his unique background in mathematics. Kumar always asked sharp, insightful questions during our meetings, which helped broaden my perspective on formulating solid research problems. I am especially indebted to Krishna, who has been an invaluable part of my graduate life. He introduced me to the beautiful tapestry of coding theory and information theory. He always took the time out for a quick chat, be it on research problems, or career advice, or Indian classical music. During our meetings, he asked stimulating questions and provided valuable feedback, which helped shape several key results in this thesis. Special thanks to Krishna (and his family) for inviting me a to take part in wonderful musical gatherings at his house.

I spent a great summer in 2013 visiting Prof. Sid Jaggi's group at the Institute of Network Coding in Hong Kong. Sid's enthusiasm for research and the energy he spends in formulating, solving, and communicating interesting research problems is infectious. Visiting his vibrant group in the early phase of my PhD provided a great momentum for my research. Thank you Sid for being an awesome host and a great Hangout/Skype collaborator.

Very special thanks go to Prof. Emina Soljanin for giving me the great opportunity to visit Bell Labs in the summer of 2014. This short but stimulating visit has resulted in an extensive collaboration, which has been a great learning experience for me. Emina is outstanding in identifying crucial engineering issues arising in practical scenarios and transforming them into sound theoretical problems. Her wizardry in presenting complex ideas in a simple form using beautiful pictures has always influenced me to reflect on how I would present my research. She is not only an exceptional researcher, but also an extraordinary mentor. Thank you Emina for your genuine care, mentorship, and encouragement.

I was fortunate to be able to have discussions with Prof. Robert Calderbank, when he visited Texas A&M University in the fall of 2015. A quest to solve an interesting problem that he posed during those discussions has turned into a continued collaboration since then. Robert's zeal to ask fundamental research questions and his uncanny ability to devise extremely elegant proofs has always inspired me to produce quality research. Thank you Robert for always taking the time out for discussions, for your lightning-fast email replies, and for inviting me to visit Duke University – it has been my privilege and my pleasure to work with you.

I got an opportunity to stay in the beautiful city of Paris for a series of workshops in the spring of 2016, which resulted in a fruitful collaboration with Prof. Iwan Duursma and Prof. Salim El Rouayheb. Thank you Iwan for always reserving time for insightful research discussions at every conference that we have met. Special thanks to Salim for many interesting discussions while taking long walks in Paris, for inviting me to Chicago as a visiting researcher, and for being my academic big-brother.

I was fortunate to serve as a Graduate Teaching Fellow of the College of Engineering for the Undergraduate Probability course for three consecutive semesters under the able mentorship of Prof. Jean-Francois Chamberland, Prof. Ulisses Braga-Neto, and Prof. Xiaoning Qian. I would like to thank them for their critical feedback as well as encouragement, which has helped me to substantially enhance my teaching skills. Special thanks to Jean-Francoise for encouraging me to try out novel teaching methods, and for checking in on me from time to time. I am grateful to my teachers here at Texas A&M University: Professors Henry Pfister (now at Duke), Srinivas Shakkottai, Catherine Yan, P.R. Kumar, and Krishna Narayanan.

I wish to thank Prof. Frank Kschischang for being a gracious host when I visited his group at the University of Toronto in the summer of 2014. I am grateful to Prof. Bikash Kumar Day for being a wonderful host when I visited Indian Institute of Technology Bombay in the summer of 2015.

I would like to thank the wonderful staff here at our department, Carolyn Warzon, Anni Brunker, Tammy Carda, Katie Bryan, Melissa Sheldon, Sheryl Mallet, and Alexandra Bello, for shielding me from all the bureaucratic nightmares.

I would like to thank all my current and past lab-mates and office-mates (at Texas A&M and outside groups that I visited): Muxi Yan, Yu-Pin Hsu, Santhosh Kumar E, Ying Wang, Ping-Chun Wang, Avinash Vem, Priya Venkat, Eric Zhang (special thanks for being a nice collaborator), Rawad Bitar, Serge Kh, Razan Tajeddinne (special thanks to these three for their hospitality in Chicago), Narayanan Rengaswami, Mengke Lian (special thanks to these two for their hospitality in Durham), Jithin Ravi, Mehmet Aktaş (special thanks for being a great Skype collaborator), Navid, and Nadia. I am grateful to a group of amazing postdocs (now research scientists) who have helped me in numerous technical as well as personal aspects: Mayank Bakshi (special thanks for being a great collaborator), Hoang Dau, and Chung Chan. Special thanks to Ankit Singh Rawat and Gauri Joshi for helping me in my preparations for an academic career. Thanks Gauri for also being an awesome collaborator.

I am especially indebted to Anoosheh Heidarzadeh, who joined our group as a postdoc in 2015 (now a research scientist). Anoosheh is a talented researcher and an amazing human being. I have had a wonderful time while attending a number of conferences with him. Anoosheh, thank you for our countless number of discussions, for entertaining my stupid questions and crazy ideas, for teaching me to be methodical and rigorous in my research approach, and for being my academic god-brother.

Life at College Station would have been cheerless without a group of incredible friends: Shamik Basu, Ankush Kothalkar, Tejas Umale, Sneha Chawla (special thanks to these for being awesome roommates), Madhavi Sinha, Yogesh Babbar, Maheshwari Neelam, Nahid Esmati (special thanks for being there to help when my parents visited USA), Samyukta Sethuraman, Sameer Jape, Swapnil Ghodge, Akshay Parchure, Abhinav Srivastava, and many others (too many to name here). I am also fortunate to have the most awesome group of friends back in India: Abhishek Kakade, Prasad P Kulkarni, Prasad G Kulkarni, Nikhil Joshi, Saurabh Pitkar, and Nilesh Jadhav, who have always been there with me through thick and thin.

Finally, and most importantly, I owe my deepest gratitude to my parents Surekha Kadhe (Aai) and Ravindra Kadhe (Baba), who have been my strongest support. Everything that I have achieved in my life, including this dissertation, would never have been possible without their unconditional love and countless sacrifices. Aai-Baba, I cannot thank you enough for being always there for me!

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Alex Sprintson (advisor) and Professors P. R. Kumar and Krishna Narayanan of the Department of Electrical and Computer Engineering and Professor J. Murice Rojas of the Department of Mathematics at the Texas A&M University. The work presented in Chapter 5 is carried out in collaboration with Professor Salim El Rouayheb of the Department of Electrical and Computer Engineering at the Rutgers University and Professor Iwan Duursma of the Department of Mathematics at the University of Illinois, Urbana-Champaign. The work presented in Chapter 6 is carried out in collaboration with Professor Emina Soljanin of the Department of Electrical and Computer Engineering at the Rutgers University.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

This work was supported in part by the National Science Foundation (NSF) under Grants No. CNS-0954153, CIF-1718658, the Air Force Office of Scientific Research (AFOSR) under contract No. FA9550-13-1-0008, and the Graduate Teaching Fellowship from the College of Engineering, Texas A&M University.

NOMENCLATURE

DSS	Distributed Storage System
MDS	Maximum Distance Separable
MRD	Maximum Rank Distance
LRC	Locally Recoverable/Repairable Code
RC	Regenerating Code
HDFS	Hadoop Distributed File System
MBR	Minimum Bandwidth Regenerating
MSR	Minimum Storage Regenerating
MR	Maximally Recoverable
PM-RC	Product-Matrix Regenerating Code
LP	Linear Program
$\mathbb{F}_q$	Finite field of order q
$\mathbb{F}_q^n$	Vector space of n-tuples over $\mathbb{F}_q$
$H(\cdot)$	q-ary Entropy
$I(\cdot;\cdot)$	Mutual Information
$[m]$	Set of integers $\{1, 2, \ldots, m\}$
$[m, n]$	Set of integers $\{m, m+1, \ldots, n\}$
$\mathsf{wt}(\mathbf{x})$	Hamming weight of a vector $\mathbf{x}$
$\mathcal{C}$	Linear code
$\mathcal{C}^{\perp}$	Dual code of a code $\mathcal{C}$
$d(\mathcal{C})$	Minimum Hamming distance of a code $\mathcal{C}$

$d_R(\mathcal{C})$	Minimum rank distance of a rank-metric code $\mathcal{C}$
$d_S(\mathcal{C})$	Minimum subspace distance of a subspace code $\mathcal{C}$
(n, k, d)	Linear code of dimension k, block-length n, and distance d
B	Number of symbols stored in a DSS
B_s	Number of symbols stored with security constraints
α	Number of symbols per node
β	Number of symbols downloaded per node during repair
d	Repair degree or minimum distance
G	Generator matrix of a code
H	Parity-check matrix of a code
$\mathsf{Rank}(H)$	Rank of a matrix H
$\mathsf{dim}(U)$	Dimension of a subspace U
r	Locality of a code
r_i	Locality of the i-th coordinate
$\mathbf{k} = \{k_1, \ldots, k_r\}$	Information locality profile of a code; k_j information symbols have locality j
$\mathbf{n} = \{n_1, \ldots, n_{r_a}\}$	All-symbol locality profile of a code; n_j symbols have locality j
$\tilde{\mathbf{k}} = \{\tilde{k}_1, \ldots, \tilde{k}_r\}$	Information locality requirement of a code; $\tilde{k}_j$ information symbols have locality at most j
$\mathcal{C}\mid_S$	Restriction of a code to the coordinates in a set S
$\mathcal{R}_l^{(i)}$	l-th repair group of the i-th node
(r, t)-Availability	Property of having at least t disjoint repair groups, each of size at most r
RV	Random Variable

PDF	Probability Density Function
CDF	Cumulative Distribution Function
$f_S(s)$	Probability Density Function of S
$F_S(s)$	Cumulative Distribution Function of S
FJ	Fork-Join (queuing) model
S_i	Service time at node i
$S^{(r,t)}$	Service time required to download a file using a code with (r, t)-availability
$T^{(r,t)}$	Total time required to download a file using a code with (r, t)-availability
$T^{(t_r)}$	Time required to download a file using a t_r-replication code
$T^{(n,k)}$	Time required to download a file using an (n, k)-MDS code

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

ACKNOWLEDGMENTS v

CONTRIBUTORS AND FUNDING SOURCES ix

NOMENCLATURE x

TABLE OF CONTENTS xiii

LIST OF FIGURES xvi

LIST OF TABLES xvii

1. INTRODUCTION 1

    1.1 Background and Motivation 1
        1.1.1 Node Repair Problem 2
        1.1.2 Regenerating Codes 3
        1.1.3 Codes with Locality and Availability 4
        1.1.4 Information-Theoretic Security in DSS 6
    1.2 Contributions and Organization 7

2. PERFECT SECURITY IN DISTRIBUTED STORAGE 12

    2.1 Introduction 12
    2.2 Preliminaries 14
        2.2.1 System Model 14
            2.2.1.1 Eavesdropper Model 15
            2.2.1.2 Information-Theoretic Security 15
        2.2.2 Secure Code Design Using a Coset Code 16
    2.3 Secure Minimum Storage Regenerating (MSR) Codes 17
        2.3.1 Background 17
        2.3.2 Framework for Secure MSR Code Design 18
    2.4 Security for Locally Repairable Codes (LRCs) 20

2.4.1 Background 20
2.4.2 Framework for Designing Secure LRCs Using Maximally Recoverable Codes 21
2.4.3 Construction of a Secure LRC Against a $(k-1,0)$-Eavesdropper .. 23
2.5 Conclusion 26

3. WEAK SECURITY IN DISTRIBUTED STORAGE 27

3.1 Introduction 27
3.2 Preliminaries 28
3.2.1 Regenerating Codes 28
3.2.2 Information-Theoretic Secrecy 30
3.3 Explicit Outer Codes for Weak Security of Product-Matrix (PM) Codes 33
3.3.1 Recap of PM Codes 33
3.3.2 Conventional Representation of PM Codes 34
3.3.3 Main Idea and Definitions 35
3.3.4 Outer Code Construction for PM-MBR Codes 37
3.4 Universally Weakly Secure Coset Codes for MSR Codes 53
3.4.1 Construction for $\ell = k-1$ and $g = 1$ 56
3.4.2 Construction for $\ell = 1$ and $g = B - \alpha$ 60
3.5 Conclusion 63

4. CODES WITH UNEQUAL LOCALITY 64

4.1 Introduction 64
4.2 Preliminaries 68
4.3 Codes with Unequal Information Locality 69
4.3.1 Bound on the Minimum Distance 70
4.3.2 Code Construction: Pyramid Codes 74
4.4 Codes with Unequal All Symbol Locality 75
4.4.1 Bound on the Minimum Distance 76
4.4.2 Code Construction Based on Rank-Metric Codes 80
4.5 Information Locality Requirement 84
4.6 Conclusion 89

5. CODES WITH LOCALITY IN THE RANK AND SUBSPACE METRICS 91

5.1 Introduction 91
5.2 Preliminaries 98
5.2.1 Linearized Polynomials 99
5.2.2 Rank-Metric Codes 100
5.2.3 Subspace Codes 103
5.2.4 Codes with Locality 105
5.3 Codes with Rank-Locality 106

5.3.1 Upper Bound on Rank Distance 107
5.3.2 Code Construction 108
5.4 Codes with Subspace-Locality 118
5.5 Correction Capability of Codes with Rank-Locality 121
5.6 Conclusion 124

6. LATENCY ANALYSIS FOR CODES WITH AVAILABILITY 126

6.1 Introduction 126
6.2 Preliminaries on Log Concavity (Convexity) of Order Statistics 131
6.3 System Model 132
6.3.1 Encoding Model 132
6.3.2 Content Access Models 133
6.4 Analysis of Download Time for Low Arrival Rate Regime 134
6.4.1 Mean Download Time 134
6.4.2 Comparison with Replication Codes 136
6.4.3 Comparison with MDS Codes 138
6.4.4 Discussion 140
6.4.5 Download Performance of Repair Groups 141
6.5 Results for the High Arrival Rate Scenario 148
6.5.1 Upper Bound on Mean Download Time 149
6.5.2 Inner Bound on Stability Region 152
6.5.3 Simulation Results 152
6.6 Service Capacity Allocation 155
6.6.1 Service Rate Allocation for No-Split Model 156
6.6.2 Service Rate Allocation for $(t+1)$-Split Model 157
6.6.3 Service Rate Allocation for 2-Split Fork Join Model 159
6.7 Conclusion 161

7. CONCLUSIONS AND FUTURE DIRECTIONS 162

7.1 Security in DSS 162
7.2 Locality in DSS 163
7.3 Availability in DSS 164

REFERENCES 166

LIST OF FIGURES

FIGURE Page

1.1 Comparison between replication and a maximum distance separable (MDS) code (over a finite field of order 5) 3

1.2 A $(5,2)$ regenerating code example. 4

1.3 A $(6,3)$ locally repairable code (LRC) example. 5

5.1 A rank-error pattern of rank one. 92

5.2 A few instances of crisscross failures affecting two rows and/or columns. .. 94

5.3 An example of a 9×9 bit array. 124

6.1 Expected service time and its upper and lower bounds as a function of t for various values of r for $\mu = 1$. 144

6.2 Probability that the repair groups download faster than the systematic node as a function of t for various values of r when $\mu = 1$. 147

6.3 Fork-join queueing system with an (n, k, r, t)-LRC. 149

6.4 Butterfly Queue: Requests for f_1 are shown in green whereas those for f_2 are shown in brown. 150

6.5 Mean download time vs. aggregate arrival rate for various coding schemes when all the files have equal popularity. 153

6.6 Mean download time vs. aggregate arrival rate for various coding schemes when 6.67% of the files have 90% of the popularity and the rest share 10% of popularity equally. 154

6.7 Three traffic flow splitting models. 155

6.8 $\mathbb{E}\left[T_{0,\text{FJ}}^{(r,t)}(\gamma)\right]$ versus γ for various values of r. 158

LIST OF TABLES

TABLE Page

6.1 Normalized mean download time for different coding approaches 138

1. INTRODUCTION

1.1 Background and Motivation

The amount of data being stored and computed on cloud systems is increasing at an enormous rate. The global cloud storage market is expected to reach $74.94 billion by 2021, with a compound annual growth rate of 25.8% [1]. Cloud services are implemented on top of a distributed storage layer that acts as a middleware to the applications, and also provides the desired content to the users. As a result, the performance of a cloud system and the quality of user experience rely on the efficiency of underlying distributed storage system. On the other hand, as the amount of information being generated is growing exponentially, the storage space is becoming a scarce resource. In addition, ensuring security of the stored information is a major concern as demonstrated by recent cyber security incidences such as Sony and Target attacks [2]. Therefore, building large-scale data centers storing exabytes of data in a secure as well as reliable manner poses a multitude of challenges.

A distributed storage system (DSS) typically stores data across a cluster consisting of hundreds of nodes connected through a networking infrastructure. Such clusters are usually built out of commodity components, which makes failures the norm rather than the exception [3, 4]. In addition, there are several systems-related issues such as server reboots, software glitches, maintenance operations, power failures etc. that cause *storage nodes* to be unavailable from time to time.

In order to enable resilience against failures, a DSS introduces some form of redundancy. Traditionally, data is replicated across multiple nodes. For instance, the Google File System [5] and the Hadoop Distributed File System (HDFS) [6] store three copies of all data by default. However, the replication strategy is expensive in terms of storage cost, especially considering the exponential growth rate of data being generated and stored. As a result, several large-scale distributed storage systems have started to resort to error correcting codes, which provide higher reliability at significantly lower storage overheads, see *e.g.*, [5, 7]. Even though classical erasure codes such as Reed-Solomon codes achieve high storage efficiency, they are inefficient in handling disk (or node) failures as they usually require to download large amount of data while repairing a failed node.

In the following, we briefly outline the challenges arising in efficiently repairing a failed storage node in a distributed storage system while maintaining high reliability and storage efficiency. Further, we briefly mention the challenges arising in guaranteeing the security of the stored data in a reliable, storage efficient, and repair efficient distributed storage system. Along the way, we give a high level overview of the related work and motivate the questions, which we are interested in tackling.

1.1.1 Node Repair Problem

Let us consider a scenario where a storage node has failed, and needs to be recovered in order to maintain the system. Fig. 1.1 compares a replication-based DSS to a maximum distance separable (MDS) coded DSS. Both the systems encode $k = 3$ blocks of data and store one (coded) block on each storage node in such a way that the system can tolerate any two node failures without any loss of the information. Notice that replication requires storing $n = 9$ blocks, whereas MDS-coded DSS stores only $n = 5$ blocks.

Now, consider a situation where one storage node, say the first node, in a DSS has failed. In the case of replication, the data on a failed node can be recovered by downloading from its replica. On the other hand, for MDS-coded DSS, it is required to contact any $k = 3$ nodes and download the entire data stored on each of them. Thus, recovering a failed node in an MDS-coded DSS is inefficient as compared to the replication, as it requires contacting k-fold more nodes and downloading k-fold more data than replication.

f_1	f_1	f_1
f_2	f_2	f_2
f_3	f_3	f_3

(a) $(9, 3)$ Replication Code

f_1	f_2	f_3	$f_1 + f_2 + f_3$	$f_1 + 2f_2 + 3f_3$

(b) $(5, 3)$ MDS code

Figure 1.1: Comparison between replication and a maximum distance separable (MDS) code (over a finite field of order 5)

.

This problem of *efficiently* repairing a failed storage node, termed as the *node repair problem*, has recently attracted significant research attention from the coding theorists. In particular, the following two metrics of repair efficiency have received particular research attention: (a) *Repair bandwidth* – the total number of symbols (or bits) communicated while repairing a failed node. The corresponding family of codes is called *regenerating codes* [8]. (b) *Repair locality* – the number of nodes participating in the repair process. The corresponding family of codes is called *locally repairable codes* [9].

1.1.2 Regenerating Codes

The class of storage codes that reduce the amount of data downloaded (*i.e.*, repair bandwidth) while repairing a node are referred to as *regenerating codes*. In their seminal work, Dimakis *et al.* [10, 8] showed that there exists a trade-off between storage space per node and repair bandwidth for a single node failure. Regenerating codes are the class of codes that optimally achieve this trade-off.

Most of the results in the literature focus on the two extreme points of the optimal storage-repair bandwidth trade-off curve. The codes on one extreme point that minimize the repair bandwidth first and then the storage per node are referred to as *Minimum Bandwidth Regenerating* (MBR) codes; whereas, the codes on the other extreme point that first minimize the storage per node and then the repair bandwidth are referred to as *Minimum*

f_{11}	f_{21}	f_{31}	$f_{11} + f_{21} + f_{31}$	$f_{11} + 2f_{23} + f_{32}$
f_{12}	f_{22}	f_{32}	$f_{12} + f_{22} + f_{32}$	$f_{12} + 2f_{24} + f_{31}$
f_{13}	f_{23}	f_{33}	$f_{13} + f_{23} + f_{33}$	$f_{13} + f_{21} + f_{34}$
f_{14}	f_{24}	f_{34}	$f_{14} + f_{24} + f_{34}$	$f_{14} + 2f_{22} + 2f_{33}$

Figure 1.2: A $(5, 2)$ regenerating code example.

Storage Regenerating (MSR) codes. Several explicit code constructions have been proposed for these extreme points considering the *exact* repair model, wherein the repaired node is an exact replica of the failed node (see *e.g.*, [11, 12, 13, 14, 15, 16] and references therein).

Fig. 1.2 gives an example of a $(5, 3)$ MSR code from [17]. The encoding is over a finite field of order 3, denoted as $\mathbb{F}_3$. Each of the three information blocks f_1, f_2, and f_3 is divided into four sub-blocks. When any of the systematic nodes fails, it can be repaired by downloading only eight sub-blocks. For instance, suppose the first node has failed. Then, its contents can be recovered by downloading sub-blocks from the first and the fourth row from nodes 2 through 4 and sub-blocks from the second and third row from node 5. Even though this code can tolerate any two node failures similar to a $(5, 3)$ MDS code, the repair bandwidth for repairing any (systematic) node is only two third as compared to the MDS code. This is proven to be information-theoretically optimal in [8]. The drawback, however, is that it is required to contact all the four remaining nodes during the repair.

1.1.3 Codes with Locality and Availability

The class of storage codes that reduce the number of nodes participating in the repair process (*i.e.*, repair locality) are known as *locally repairable (recoverable) codes* (LRCs). The set of nodes participating in the repair are called *repair group* of the node being recovered. Codes with small locality were introduced in [18, 19] (see also [20]). The study of the locality property was galvanized with the pioneering work of Gopalan *et*

f_1	f_2	f_3	$f_1 + f_2$	$f_1 + f_3$	$f_2 + f_3$

Figure 1.3: A $(6, 3)$ locally repairable code (LRC) example.

al. [9]. One of their key contributions was to establish a trade-off between the minimum distance of a code and its locality, analogous to the classical Singleton bound. Since then, a series of results have extended the code distance bound for various types of codes along with corresponding *optimal* code constructions achieving the minimum distance bound (see, *e.g.*, [21, 22, 23, 24, 25, 26], and references therein).

Fig. 1.3 depicts an example of a $(6, 3)$ LRC with locality of two for every node. In other words, any node can be repaired by contacting two other nodes. For instance, f_1 can be recovered from f_2 and $f_1 + f_2$. Note that this is only two third of the nodes required to be contacted for any MDS code that stores three data blocks. The minimum distance of this code is 3, which is proven to be optimal amongst all linear codes with dimension 3 and locality 2 (see [9]).

In this thesis, we strive to seek answers to the following questions regarding LRCs.

1. In conventional LRCs, every node usually has the same locality. Can one design codes that enable smaller locality for nodes storing important symbols? How would such a property affect the trade-off between locality and minimum distance (fault tolerance)?

2. In the literature, locality requirements are considered under the Hamming metric. Can we generalize the notion of locality to the *rank metric*, wherein codewords are matrices and the distance between two codewords is the rank of their difference? How to design codes that can optimally achieve the trade-off between locality and minimum distance in the rank metric?

If a node can be repaired from multiple, disjoint repair groups of small size, then it is said to have the *availability* property. Availability is particularly attractive for downloading frequently accessed data, called *hot data*, as it enables a number of ways to parallelly download the data. For example, the code depicted in Fig. 1.3 has availability of two, *i.e.*, any node can be repaired from two disjoint repair groups, each of size two. For instance, f_1 can be repaired from either f_2 and $f_1 + f_2$, or f_3 and $f_1 + f_3$. It has been shown that there is a trade-off between the availability and the minimum distance of the code, and also between the availability and the rate of the code (see *e.g.*, [27, 28, 29, 30], and references therein).

We are interested in finding answers to the following questions regarding LRCs with availability.

1. Does coding theoretic availability guarantee fast access when service time at a node is random?

2. How do the codes with availability perform in terms of download delay in comparison with conventional schemes?

1.1.4 Information-Theoretic Security in DSS

Another important issue that is crucial for successful implementation of a DSS is its resilience to adversarial attacks. Due to their decentralized nature, DSS are susceptible to adversarial attacks by malicious users and/or network operators. In such attacks, an adversary may attempt to gain access to the valuable information stored on the system or may want to modify the stored information in order to disrupt the system. We restrict our attention to passive eavesdroppers that can observe the data stored on limited number of storage nodes.

Conventional secret-key-based encryption techniques require sophisticated secret-key management mechanisms, which incur significant computational and communication overheads in distributed settings. This has motivated us to focus on the paradigm of information-theoretic security. Intuitively, perfect information-theoretic security requires that the eavesdropper gains absolutely *no information* about the stored data from its observations. Following the work of [31, 32], a number of investigations have been carried out on information-theoretically securing the regenerating codes (see, *e.g.*, [33, 34, 22]).

For many practical storage systems, the perfect secrecy condition might be too strong. Moreover, coding schemes that provide perfect secrecy involve mixing data symbols with random keys to confuse the eavesdropper, which incurs loss in the storage capacity. In view of this, an interesting model of security, called *weak security*, was introduced by Bhattad and Narayanan in [35]. The intuition behind the weak security is to protect individual and small groups of symbols from the eavesdropper. Due to its practical benefits, the weak security model has started receiving attention for distributed storage applications, see for instance, [36, 37, 38].

We ask the following questions regarding secure DSS.

1. Secure constructions of repair efficient codes in the literature typically require the size of the underlying field to be large. How can we design information theoretically secure regenerating and locally repairable codes over small field size?

2. How can we construct small field size schemes for weak security?

1.2 Contributions and Organization

In a nutshell, designing DSS that guarantees security of cloud systems, ensures reliability against failures, and provides high availability for retrieving partial information poses numerous challenges. These conflicting requirements of security, reliability, storage efficiency, and repair efficiency in data centers have created a new set of problems for coding and information theorists. In this thesis, we address several of the challenges as elaborated in the following.

In **Chapter 2**, we consider the problem of designing repair efficient distributed storage systems, which are information-theoretically secure against a passive eavesdropper that can gain access to a limited number of storage nodes. We present a framework that enables design of a broad range of secure storage codes through a joint construction of inner and outer codes. As case studies, we focus on two specific families of storage codes: (i) *minimum storage regenerating* (MSR) codes, and (ii) *maximally recoverable* (MR) codes, which are a sub-class of *locally repairable codes* (LRCs). The main idea of this framework is to utilize the existing constructions of storage codes to jointly design an outer coset code and inner storage code. Finally, we present a construction of an outer coset code over small field size to secure locally repairable codes presented by Tamo and Barg [25] for the special case of an eavesdropper that can observe any subset of nodes of maximum possible size. This scheme can be considered as a secret sharing scheme with local recoverability for shares, wherein a lost share can be recovered from a small set of other shares.

In **Chapter 3**, our focus is on designing outer codes to achieve weak security in a DSS that employs regenerating codes. First, we focus on a well-known family of regenerating codes called Product-Matrix (PM) codes [13], and present coset coding based outer code constructions to weakly secure PM codes when eavesdropper can observe any single storage node. The key benefit of the proposed precoding schemes is that they require smaller alphabet size (in particular, linear in number of storage nodes) as compared to the existing code constructions for weak security [39, 38] (which require alphabet size that is exponential in number of storage nodes). Next, we construct *universal* outer codes that can weakly secure *any* MSR code against an eavesdropper observing any single node. Our schemes allow the independent design of the storage code and the outer code. The key benefit of the proposed schemes is their smaller field size than that of the existing universal outer codes.

In **Chapter 4**, inspired by the notion of *unequal error protection*, we investigate linear codes, in which, different nodes possess different values of locality. We refer to such codes as *codes with unequal locality*. First, we consider a class of codes with *unequal information locality*, *i.e.*, systematic codes with unequal locality constraints imposed only on the information symbols. For this class of codes, we compute a tight upper bound on the minimum distance as a function of locality constraints. We demonstrate that the construction of Pyramid codes by Huang *et al.* [18] can be adapted to design *optimal* codes with unequal information locality that achieve the minimum distance bound.

Next, we consider codes with *unequal all-symbol locality*, *i.e.*, codes in which the locality constraints are imposed on all symbols. We establish an upper bound on the minimum distance as a function of number of symbols of each locality value. We show that the construction based on rank-metric codes by Silberstein *et al.* [24] can be adapted to obtain optimal codes with unequal all-symbol locality.

Finally, we introduce the concept of *locality requirement* of a code, which can be viewed as a recoverability requirement on symbols. Information locality requirement of a code essentially specifies the minimum number of information symbols of each locality value that must be present in the code. For a given locality requirement, we present a greedy algorithm to construct codes that have maximum minimum distance among all codes that satisfy the locality requirement.

In **Chapter 5**, we construct codes with locality constraints in the rank and subspace metrics (instead of the conventional Hamming metric). Our motivation stems from designing codes for efficient data recovery from correlated and/or mixed (i.e., complete and partial) failures in distributed storage systems. Specifically, the proposed local rank-metric codes can recover locally from *crisscross erasures and errors*, which affect a limited number of rows and/or columns of the storage system. We prove a Singleton-like upper bound on the minimum rank-distance of linear codes with *rank-locality* constraints. Then, we construct a family of locally recoverable rank-metric codes that achieve this bound for a broad range of parameters. The proposed construction builds upon Tamo and Barg's method for constructing locally repairable codes with optimal minimum Hamming distance [25]. Finally, we construct a class of subspace codes (also known as codes in projective space) with locality constraints in the subspace metric. The key idea is to show that a subspace code with locality can be easily constructed from a rank-metric code with locality by using the lifting method proposed by Silva *et al.* One potential application of such codes is for distributed storage systems, wherein nodes are connected over a network that can introduce errors and erasures.

In **Chapter 6**, we study codes with availability (referred to as availability codes) from a queuing-theoretical perspective. We first analyze the impact of availability parameters under the assumption that the download requests arrive at a low rate. In particular, we investigate how the size and the number of repair groups affect the download latency by computing the mean download time when using multiple repair groups and the probability that repair groups perform slower download than the systematic node. Next, for high arrival rate regime, we analyze the average time necessary to download a block of data under the Poisson request arrival model in two service/scheduling scenarios. We compare the availability codes with several alternatives such as MDS codes and replication schemes. Our results indicate that availability codes can minimize the download time for certain request arrival patterns, but are not always optimal in terms of minimizing the download time.

Then, we explore the case when the cumulative service capacity of the system is limited, and determine the optimal allocation of the service capacity across nodes such that mean download delay of a single request is minimized. We consider this resource allocation problem under three traffic splitting models, which differ in what fraction of requests are routed to the node storing the desired data block and to one or more of its repair groups. Here, we assume that the download requests arrive at a low rate. We find that although, in principle, higher availability should help in reducing download delays, this is not the case in our scenario.

Finally, in Chapter 7, we summarize our results, and present further directions.

2. PERFECT SECURITY IN DISTRIBUTED STORAGE*

2.1 Introduction

Distributed storage systems (DSS) are required to provide reliability in the face of machine failures, and security in the presence of malicious users. Reliability is typically obtained by adding some form of redundancy via replication or erasure coding. The enormous growth of the amount of data being stored and computed online has motivated practical systems to use erasure codes for handling failures. This has galvanized a significant amount of work in the past few years on novel erasure codes that efficiently handle node failures in DSS (see, *e.g.*, [8, 13, 9, 25]).

Another important challenge for a DSS is the security of the stored data. Due to their decentralized nature, DSS are susceptible to eavesdropping attacks by malicious users, in which an eavesdropper may attempt to gain access to the valuable information stored.

In this chapter, we are interested in information-theoretic security of DSS against passive eavesdroppers that can gain access to a limited number of storage nodes. We restrict our attention to DSS that employ *repair efficient* erasure codes, focusing on the following two families of codes: (a) *regenerating codes* (RC) that minimize repair bandwidth, *i.e.*, the amount of data downloaded while repairing a failed node (see, *e.g.*, [8]); and (b) *locally repairable codes* (LRC) that minimize locality, *i.e.*, the number of nodes participating in the repair process (see, *e.g.*, [9]).

Information-theoretic security requires that an eavesdropper accessing a limited number of storage nodes cannot gain *any information* about the stored data. Specifically, suppose a DSS encodes a set S of B_s symbols (over a finite field $\mathbb{F}_q$) using an RC or an LRC

*Reprinted with permission from [40] "Security for minimum storage regenerating codes and locally repairable codes," by S. Kadhe and A. Sprintson, 2017. *In Proceedings of 2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 1028-1032, June 2017. Copyright © by IEEE.

into a set of symbols C, stored across n nodes. Consider an eavesdropper, *Eve*, who can observe data stored on any ℓ_1 nodes, and in addition, can observe data downloaded during the repair of any ℓ_2 nodes. We refer to Eve as an (ℓ_1, ℓ_2)-eavesdropper. Let $C(E)$ denote the set of (encoded) symbols that Eve observes. A DSS is said to achieve information-theoretic security against Eve if the mutual information between the information symbols S and the eavesdropped symbols $C(E)$ is zero, *i.e.*, $I\left(S; C(E)\right) = 0$.

Information-theoretic security for RCs and LRCs has recently received a significant attention from the research community, see *e.g.*, [32, 33, 41, 22, 42, 43]. The focus has been on designing secure coding schemes, and on characterizing outer bounds on the amount of data B_s that can be securely stored.

A secure coding scheme usually *mixes* the information symbols to be stored with random symbols, which are independent of the information symbols. In the secure distributed storage literature, one can find two ways of mixing random symbols: (i) first precode information symbols and random symbols using a *maximum rank distance* (MRD) code such as a Gabidulin code [44], and then use a storage code to encode the precoded symbols [22, 42, 43]; or (ii) directly mix information and random symbols using a storage code [33, 42]. In the latter case, information symbols are usually appended to *random symbols* (more generally, information and random symbols are interleaved), and then encoded together using a storage code.

Even though precoding by an MRD code can be used to secure a large class of storage codes, its main drawback is that the required field size is exponential in the number of storage nodes. On the other hand, when directly mixing information and random symbols, field size is governed by the storage code. However, it is not clear if it is always possible to achieve security by directly mixing information and random symbols.

In this chapter, we are interested in designing secure coding schemes by using an outer code based on coset coding [45] to mix information and random symbols[2], and then use an RC or LRC as an inner code. Our goal is to design coding schemes over small field. Towards this, our approach is to design a coset code by leveraging the structure of the underlying storage code. We outline our contributions in the following.

Our contributions: First, we focus on a class of RCs, namely, minimum storage regenerating (MSR) codes, and present a framework to design a coset code for security against an $(\ell_1, 0)$-eavesdropper by utilizing a non-secure MSR code of smaller rate and low *repair degree*. Next, we consider security for LRCs, and extend our framework to construct coset codes for security against an (ℓ_1, ℓ_2)-eavesdropper by utilizing a class of LRCs called *maximally recoverable* (MR) codes [46]. As we demonstrate in Remarks 3 and 5, the proposed framework enables us to leverage existing constructions of MSR and MR codes with little additional cost to construct secure codes. Finally, we present a construction of a coset code over field size of $O(n^{k/r})$ to secure LRCs with locality r and block-length n presented in [25] against a $(k-1, 0)$-eavesdropper.

2.2 Preliminaries

Notation: For an integer m, let $[m] = \{1, 2, \ldots, m\}$, and for a pair of integers $m < n$, let $[m, n] = \{m, m+1, \ldots, n\}$. For an $n \times m$ matrix G and a set $A \subset [n]$, let G_A denote a $|A| \times m$ sub-matrix of G containing the rows of G that are indexed by A.

2.2.1 System Model

Consider a DSS that encodes a set of B symbols $X \in \mathbb{F}_q^B$ into $n\alpha$ symbols $C \in \mathbb{F}_q^{n\alpha}$ as $C = GX$, where $G \in \mathbb{F}_q^{n\alpha \times B}$ is a generator matrix of an erasure code, which we refer to as a *storage code*. These $n\alpha$ symbols are stored across n nodes, with each node storing α symbols. We denote the symbols stored on node i as $C_i = G_i X$, where G_i is an $\alpha \times B$

[2] We note that directly mixing information and random symbols can also be considered as coset coding. We elaborate the connections in Sec. 2.2.2.

sub-matrix of G. When a node fails, its contents are recovered by downloading symbols from a subset of the remaining $n-1$ nodes. The specifics of this repair process depend on the storage code.

We focus on two families of codes that facilitate efficient repair process, namely, regenerating codes (RCs) and locally repairable codes (LRCs). Our objective is to design outer coset codes for RCs and LRCs, which ensure *security* against an eavesdropper that can gain access to a limited number of storage nodes. We will briefly describe RCs and LRCs in subsequent sections before discussing their secure designs. We begin with the eavesdropper model in the following.

2.2.1.1 Eavesdropper Model

We consider an (ℓ_1, ℓ_2)-eavesdropper model introduced in [33] (see also [22]). An (ℓ_1, ℓ_2)-eavesdropper, *Eve*, can access the contents of any set E_1 of nodes of size up to ℓ_1. In addition, Eve can also observe the data downloaded during the repair of any set E_2 of nodes of size up to ℓ_2. Note that Eve can potentially gain more information by observing the data downloaded during node repair than merely observing the data stored on the node, since the stored data is a function of the data downloaded during the repair.

2.2.1.2 Information-Theoretic Security

Our goal is to guarantee that *no information* about the stored data is leaked to Eve. In particular, let S be the set of information symbols, C_i be the set of symbols stored on the i-th node, and C^i be the set of symbols downloaded during the repair of node i. Further, for a set $A \subseteq [n]$, let $C_A = \{C_i : i \in A\}$, and $C^A = \{C^i : i \in A\}$.

Definition 1. *A DSS is said to achieve information-theoretic security against an (ℓ_1, ℓ_2)-eavesdropper if we have*

$$I\left(S; C_{E_1}, C^{E_2}\right) = 0, \tag{2.1}$$

for all $E_1, E_2 \subset [n]$ *such that* $|E_1| \leq \ell_1$ *and* $|E_2| \leq \ell_2$.

The maximum number of symbols B_s that a DSS can store with information-theoretic security against an (ℓ_1, ℓ_2)-eavesdropper is called the *secrecy capacity* of the DSS.

2.2.2 Secure Code Design Using a Coset Code

Our goal is to store a set S of B_s symbols securely in a DSS, which can store B symbols without any security requirements. This is typically done by *mixing* the B_s symbols of S with $B - B_s$ *random symbols* $R = \{R_1, \ldots, R_{B-B_s}\}$, where every random symbol R_i is drawn uniformly at random over $\mathbb{F}_q$, independently of S and other random symbols. We use a coset coding based outer code to first *mix* the random and information symbols, and then use an RC or LRC to encode the precoded symbols.

Outer Code Based on Coset Coding: A coset code is constructed using a $(B, B-B_s)$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ with parity-check matrix $H \in \mathbb{F}_q^{B_s \times B}$ [45]. Specifically, information symbols $S \in \mathbb{F}_q^{B_s}$ are first encoded by selecting $X \in \mathbb{F}_q^{B}$ uniformly at random such that $S = HX$. In other words, the vector S can be considered as a syndrome specifying a coset of $\mathcal{C}$, and the codeword X is a randomly chosen element of that coset.

Next, the codeword X is encoded using a storage code (as an inner code) to obtain $C \in \mathbb{F}_q^{n\alpha}$, *i.e.*, $C = GX$, where $G \in \mathbb{F}_q^{n\alpha \times B}$ is a generator matrix of the storage code. To obtain the information symbols S, a user needs to first decode the storage code to get X, and then, decode the outer coset code to get S. The decoding operation of a coset code consists of simply computing the syndrome $S = HX$. The node repair process is inherited from the storage code (which is an RC or an LRC).

To design the matrix H appropriately, we need to write the information-theoretic security condition (2.1) in terms of coding matrices H and G. Towards this, we consider the following result from [47, Lemma 6], [48, Theorem 1].

Lemma 1. *([47]) Suppose a set of information symbols S is encoded using a coset code with parity-check matrix $H \in \mathbb{F}_q^{B_s \times B}$ and a storage code with generator matrix $G \in \mathbb{F}_q^{n\alpha \times B}$. Let $C(E) = TX$ be a set of μ $(\leq B - B_s)$ symbols observed by an eavesdropper, where T is a $\mu \times B$ sub-matrix of G. Then, we have*

$$\text{rank}\, H + \text{rank}\, T = \text{rank} \begin{bmatrix} H \\ T \end{bmatrix} \Longrightarrow I\left(S; C(E)\right) = 0. \tag{2.2}$$

Our central approach is to design a coset coding matrix H by leveraging the properties of G in order to ensure the *rank condition* in (3.7).

Remark 1. *Note that appending information symbols to random symbols (in general, interleaving information and random symbols), and directly encoding them using a storage code $\mathcal{C}$ can be considered as performing coset coding using a subcode of $\mathcal{C}$. In the terminology above, this corresponds to the case where H is a sub-matrix of an identity (in general, permutation) matrix. On the other hand, for a given S, choosing $X \in \mathbb{F}_q^B$ uniformly at random such that $S = HX$ is equivalent to mixing S with uniform random symbols R as $X = \begin{bmatrix} H' \\ H \end{bmatrix}^{-1} \begin{bmatrix} R \\ S \end{bmatrix}$, where $H' \in \mathbb{F}_q^{B - B_s \times B}$ is a matrix such that $\begin{bmatrix} H' \\ H \end{bmatrix}$ is non-singular.*

2.3 Secure Minimum Storage Regenerating (MSR) Codes

2.3.1 Background

Regenerating codes (RCs) are a class of codes that minimize the amount of data downloaded to repair a failed storage node [8]. A regenerating code encodes a set of B symbols $X \in \mathbb{F}_q^B$ into $n\alpha$ symbols $C \in \mathbb{F}_q^{n\alpha}$, and stores them over n nodes with each node storing α symbols such that the system satisfies the following two properties: (i) *Reconstruction property* – connecting to *any* k out of n nodes is sufficient to reconstruct the entire stored data; (ii) *regeneration property* – when a storage node fails, its content can be *exactly regenerated* by downloading β symbols each from any d out of the remaining $n-1$ nodes. The parameter d is referred to as the *repair degree*, and the $d\beta$ number of symbols downloaded is referred to as the *repair bandwidth*. An RC with these parameters is referred to as an (n, k, α, d, β) RC.

Dimakis *et al.* [8] showed that there exists a trade-off between storage space per node (α) and repair bandwidth $(d\beta)$. We restrict our attention to *Minimum Storage Regenerating* (MSR) codes, which achieve an extreme point of the optimal storage-repair bandwidth trade-off curve corresponding to minimum storage per node.

The problem of analyzing secrecy capacity of DSS employing MSR codes, and designing secure coding schemes has been considered in [33, 41, 22, 43], and references therein. Shah *et al.* [33] presented secure codes that directly mix information and random symbols using product-matrix based MSR constructions in [13], for which a field size of $q \geq n(d-k+1)$ is sufficient. However, these secure schemes are of low rate (in particular, rate lower than half), since product-matrix MSR codes are themselves low rate codes. Secure MSR codes based on precoding by Gabidulin codes are presented in [22, 43]. Even though these schemes cover a wide range of parameters, the required field size is exponential in n due to Gabidulin precoding. We are interested in constructing secure high rate codes over small field size.

2.3.2 Framework for Secure MSR Code Design

In this section, we restrict our attention to the case $\ell_2 = 0$. For this case, the secrecy capacity of MSR codes has been shown to be $B_s = (k - \ell_1)\alpha$ (see [32, 33]). Given positive integers n, k, α, d, β, and ℓ_1 such that $k < n$, $k \leq d < n$, and $\ell_1 < k$, we present a framework to jointly design a $(k\alpha, (k-\ell_1)\alpha)$ coset code and an (n, k, α, d, β) MSR code that can store $B_s = (k - \ell_1)\alpha$ symbols securely against an $(\ell_1, 0)$-eavesdropper. The framework relies on the existence of an $(n', k, \alpha, d, \beta)$ MSR code such that $n' \geq n+k-\ell_1$ and $d \leq n-1$.

Construction 1. *Let $\mathcal{C}'$ be an $(n', k, \alpha, d, \beta)$ MSR code over $\mathbb{F}_q$ such that $n' \geq n+k-\ell_1$ and $d \leq n-1$. Let $\mathcal{G}' = \left[G_1^T \; G_2^T \; \cdots \; G_{n'}^T\right]^T$ be an $n'\alpha \times k\alpha$ generator matrix of $\mathcal{C}'$, where G_i, $1 \leq i \leq n'$, denotes an $\alpha \times k\alpha$ sub-matrix of G' corresponding to symbols stored on node i. We call sub-matrix G_i, $1 \leq i \leq n'$, as the i-th thick-row of $\mathcal{G}'$, and we refer to $\mathcal{C}'$ as a parent code. Let H be the first $k-\ell_1$ thick-rows of $\mathcal{G}'$, and G be the next n thick-rows of $\mathcal{G}'$, i.e., $H = \mathcal{G}'_{[(k-\ell_1)\alpha]}$, and $G = \mathcal{G}'_{[(k-\ell_1)\alpha+1,(n+k-\ell_1)\alpha]}$.*

To encode a set of $B_s = (k-\ell_1)\alpha$ symbols $S \in \mathbb{F}_q^{B_s}$, use a coset code based on H and a storage code generated by G as described in Section 2.2.2.

It is easy to show that the above construction results in a secure MSR code as follows.

Theorem 1. *Construction 1 yields an (n, k, α, d, β) MSR code $\mathcal{C}$ that is secure against an $(\ell_1, 0)$-eavesdropper.*

Proof: Since $\mathcal{C}'$ is an MSR code and $d \leq n-1$, $\mathcal{C}$ is also an MSR code. To prove security, note that an $(\ell_1, 0)$-eavesdropper observes $C(E) = TX$, where T consists of ℓ_1 thick-rows of G corresponding to the eavesdropped nodes. Now, since $\mathcal{C}'$ is an (n', k)-MSR code, $\begin{bmatrix} H \\ T \end{bmatrix}$ is non-singular. This is because any k thick-rows of a generator matrix of an (n', k) MSR code form an invertible matrix, since one can reconstruct the entire data from any k nodes. Hence, the scheme is secure by Lemma 2 (see (3.7)). ■

Remark 2. *Note that we can take any $k-\ell_1$ thick-rows of $\mathcal{G}'$ as H and any n of the remaining $n'-k+\ell_1$ thick-rows as G. Further, if $\mathcal{G}'$ is in systematic form, and if we use a sub-matrix of the identity matrix as H, coset coding is equivalent to appending the information symbols to random symbols.*

Remark 3. *Construction 1 enables us to construct secure MSR codes by utilizing existing MSR codes. For example, we can leverage a recent construction by Ye and Barg [16, Sec. VII] for MSR codes over $\mathbb{F}_q$ for any n, k, and d such that $k \leq d \leq n-1$ and $q \geq n+1$. Using such an (n', k) code as a parent code with $n' = n + k - \ell_1$ and $d \leq n-1$, we can construct an (n, k) MSR code that is secure against an $(\ell_1, 0)$-eavesdropper for any n, k, and d such that $k \leq d \leq n-1$ and $\ell_1 < k$ over any field of size $q \geq n + k - \ell_1 + 1$. To the best of our knowledge, this is the first construction of secure high rate MSR codes over a field of size linear in n.*

Remark 4. *Construction 1 can be used to secure a larger class of codes, namely Maximum Distance Separable (MDS) array codes. In fact, MSR codes are MDS array codes with optimum repair bandwidth. We note that the idea of splitting a longer Reed-Solomon code (which is a scalar MDS code) to jointly design H and G was used to construct secure network codes for combination networks in [48].*

2.4 Security for Locally Repairable Codes (LRCs)

2.4.1 Background

Locally repairable codes (LRCs) are a class of codes that minimize the number of nodes participating in the repair of a failed storage node [9]. An LRC encodes a set of k symbols $X \in \mathbb{F}_q^k$ into n symbols $C \in \mathbb{F}_q^n$ such that for any symbol C_i, there exists a set $\mathcal{R}_i \subset [n] \setminus \{i\}$ with $|\mathcal{R}_i| \leq r$ such that C_i can recovered from the symbols indexed by $\mathcal{R}_i$. The parameter r is referred to as the *locality* of the code, and the set $\mathcal{R}_i$ is referred to as a *repair group* of the i-th symbol C_i. An LRC with these parameters is referred to as an (n, k, r) LRC.[3] In [9], it is shown that the minimum distance d_{min} of an (n, k, r) LRC is

[3] Throughout this section, we consider scalar LRCs where each node stores $\alpha = 1$ symbol. Our results can be generalized for vector (array) LRCs with $\alpha > 1$.

upper bounded as

$$d_{\text{min}} \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{2.3}$$

In [22, 42], an upper bound on the number of symbols B_s that an (n, k, r) LRC can securely store against an (ℓ_1, ℓ_2)-eavesdropper has been shown to be

$$B_s \leq m - \left\lfloor \frac{m}{r+1} \right\rfloor - \ell_1 - r\ell_2, \tag{2.4}$$

where $m = n - d_{\text{min}} + 1$. Note that from any m out of n symbols the entire stored data can be obtained. It is easy to verify that for codes achieving equality in (2.3), equation (2.4) reduces to $B_s \leq k - \ell_1 - r\ell_2$. The authors of [22] present an achievable scheme using Gabidulin precoding, which makes the required field size to be exponential in n. The authors of [42] present a secure coding scheme that directly mixes data and random symbols using an LRC in [25]. Though the scheme operates over a small field size that is linear in n, it ensures security only against an $(\ell_1, 0)$-eavesdropper such that $\ell_1 \leq r$.

In the following section, we extend Construction 1 to obtain secure LRCs using a parent *maximally recoverable* code [46].

2.4.2 Framework for Designing Secure LRCs Using Maximally Recoverable Codes

We begin with a definition of maximally recoverable codes [46]. We assume that $(r+1) \mid n$ and $r \mid k$.

Definition 2. *Consider an (n, k, r) LRC $\mathcal{C}$. Let $\mathcal{P} = \{P_1, P_2, \cdots, P_{n/(r+1)}\}$ be a partition of $[n]$ into $n/(r+1)$ sets of size $r+1$ such that for any $i \in P_j$, the repair group of the i-th symbol in $\mathcal{C}$ is $\mathcal{R}_i = P_j \setminus \{i\}$, for every $j \in [n/(r+1)]$. The code $\mathcal{C}$ is called maximally recoverable (MR) if the code obtained by puncturing any one symbol from every P_j is an $(nr/(r+1), k)$ MDS code. We refer to such a code as an $(n, k, r, \mathcal{P})$ MR code. We refer to each P_i as a local group.*

Next, we extend Construction 1 to obtain secure LRCs using a parent LRC that is maximally recoverable. Specifically, given positive integers n, k, r, ℓ_1 and ℓ_2 such that $r < k < n$ and $\ell_1 + r\ell_2 < k$, we present a framework to jointly design a $(k, k - \ell_1 - r\ell_2)$ coset code and an (n, k, r) LRC that can store $B_s = k - \ell_1 - r\ell_2$ symbols securely against an (ℓ_1, ℓ_2)-eavesdropper. The framework relies on the existence of an $(n', k, r, \mathcal{P})$ MR code such that $n' \geq n + \lceil (k - \ell_1 - r\ell_2)/r \rceil (r + 1)$ and $(r + 1) \mid n'$.

Construction 2. *Let $\mathcal{C}'$ be an $(n', k, r, \mathcal{P})$ MR code over $\mathbb{F}_q$ such that $n' \geq n + \lceil (k - \ell_1 - r\ell_2)/r \rceil (r + 1)$ and $(r + 1) \mid n'$. Let $\mathcal{G}'$ be an $n' \times k$ generator matrix of $\mathcal{C}'$. Partition $\mathcal{G}'$ into $n'/(r + 1)$ sub-matrices $G'_1, G'_2, \ldots, G'_{n'/(r+1)}$ such that $G'_i = \mathcal{G}'_{P_i}$. We denote the j-th row in G'_i as $G'_{i,j}$. We refer to $\mathcal{C}'$ as a parent code.*

Let $u = \lceil (k - \ell_1 - r\ell_2)/r \rceil$, and $v = (k - \ell_1 - r\ell_2) \mod r$. Let $H = \mathcal{G}'_A$, where $A = \{i, j : j \in [r]$ when $i \in [u - 1]$ and $j \in [v]$ when $i = u\}$, and $G = \mathcal{G}'_B$, where $B = [u(r + 1) + 1, (u + n)(r + 1)]$.

To encode a set of $B_s = k - \ell_1 - r\ell_2$ symbols $S \in \mathbb{F}_q^{B_s}$, use a coset code based on H and a storage code generated by G as described in Section 2.2.2.

Note that H is essentially obtained by taking the rows from $\mathcal{G}'$ corresponding to the first u local groups, and deleting one row each from the first $u - 1$ local groups, while deleting $r + 1 - v$ rows from the last one.

Theorem 2. *Construction 2 yields an (n, k) MR code $\mathcal{C}$ with locality r that is secure against an (ℓ_1, ℓ_2)-eavesdropper.*

Proof: Since $\mathcal{G}'$ generates an MR code with locality r, the code generated by G is also MR with locality r. Moreover, security of $\mathcal{C}$ also follows from the fact that the parent code $\mathcal{C}'$ is MR. Specifically, let $E \subset [n]$ be the set of indices of symbols observed by Eve. Note that if Eve observes a node i during its repair, it effectively observes all the nodes in the repair group of i. Thus, we have $|E| \leq \ell_1 + r\ell_2$. Now, label the rows of G from 1 to n, and let G_E be the sub-matrix of G containing the rows indexed by E. Let $E' \subseteq E$ such that $G_{E'}$ is a full-rank sub-matrix of G_E. Then, as $\mathcal{G}'$ generates an MR code, $\begin{bmatrix} H \\ G_{E'} \end{bmatrix}$ must have full row rank. This guarantees security using Lemma 2 (see (3.7)). ■

Remark 5. *Construction 2 enables us to construct secure MR codes by leveraging existing constructions of MR codes. A number of constructions of MR codes for a wide range of parameters are presented in [46, 49]. For instance, reference [49, Construction A.2] presents constructions of* (n, k, r) *MR codes over a field of size* $q = n^{m+h-\lceil h/n \rceil -1}$*, where* $m = n/(r+1)$ *and* $h = nr/(r+1) - k$*, when* m *and* $r+1$ *are powers of a prime. By using such an* (n', k) *MR code as a parent code with smallest* n' *such that* $n' \geq n + \lceil (k - \ell_1 - r\ell_2)/r \rceil (r+1)$*,* $(r+1) \mid n'$*, and* $n'/(r+1)$ *is a power of a prime, we can construct a secure* (n, k, r) *MR code over a field of size* $(n')^{m'+h'-\lceil h'/n' \rceil -1}$*, where* $m' = n'/(r+1)$ *and* $h' = n'r/(r+1) - k$*. One can verify that, for a broad range of parameters, this gives a smaller field size than* n^k*, required by Gabidulin precoding of LRCs operating over a field of size* n*.*

Remark 6. *In [42], it is shown that it is possible to achieve security against an* $(\ell_1, 0)$*-eavesdropper by directly mixing information and random symbols using an LRC* $\mathcal{C}$*, if there exists a subcode of* $\mathcal{C}$ *of dimension* ℓ_1 *that is maximally recoverable. Note that, in general an LRC* $\mathcal{C}$ *may not contain a subcode of dimension* ℓ_1 *that is MR, even if* $\mathcal{C}$ *itself is an MR code.*

2.4.3 Construction of a Secure LRC Against a $(k-1, 0)$-Eavesdropper

In this section, we construct a coset code to secure LRCs presented in [25] for the special case of a $(k-1, 0)$-eavesdropper. Note that for a code of dimension k, we should have $\ell_1 \leq k-1$ in order to achieve non-zero secrecy capacity. For $\ell_1 = k-1$, we get $B_s = 1$ from (2.4), as LRCs in [25] achieve equality in (2.3). This special case of $\ell_1 = k-1$ can be useful in secret sharing setup as we mention in Remark 9. As in the previous section, we assume $(r+1) \mid n$ and $r \mid k$.

First, we briefly summarize the LRC construction from [25], which we refer to as Tamo-Barg (TB) codes. Let $A = \{\alpha_i : i \in [n]\} \subset \mathbb{F}_q$ be a set of n points, called *evaluation points*. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_{n/(r+1)}\}$ be a partition of A such that $|A_i| = r+1$ for each $i \in [n/(r+1)]$. Let $h(x)$ be a polynomial of degree $r+1$, called a *good polynomial*, such that it is constant on A_i for every $i \in [n/(r+1)]$. For a given set of information symbols $M \in \mathbb{F}_q^k$, a codeword is obtained by evaluating $g_M(x) = \sum_{i=0}^{k-1} M_i x^{i \bmod r} h(x)^{\lfloor i/r \rfloor}$ on the points of A. It is shown that, if A_i's are cosets of a multiplicative subgroup of $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$, then $h(x) = x^{r+1}$ can be used as a good polynomial. In this case, we can write the (i,j)-th entry of the generator matrix G of $\mathcal{C}$ as

$$G_{i,j} = \alpha_i^{[(j-1) \bmod r] + (r+1)\lfloor (j-1)/r \rfloor}, \qquad (2.5)$$

where $i \in [n]$ and $j \in [k]$.

Next, we present a construction of a coset code for securing TB codes against a $(k-1, 0)$-eavesdroppper.

Construction 3. *Let $\mathcal{C}$ be an (n,k,r) TB code obtained by choosing evaluations points $A \subset \mathbb{F}_q$, $q \geq n$, such that A_i's are cosets of a multiplicative subgroup of $\mathbb{F}_q^*$, with good polynomial $h(x) = x^{r+1}$. Let H be a $1 \times k$ matrix with j-th entry as*

$$H_{1,j} = \beta^{[(j-1) \bmod r] + (r+1)\lfloor (j-1)/r \rfloor}, \qquad (2.6)$$

where β is a primitive element of $\mathbb{F}_Q$ with $Q = q^{k/r}$. Note that $\mathbb{F}_Q$ is an extension field of $\mathbb{F}_q$.

To encode an information symbol $S \in \mathbb{F}_Q$, use a coset code based on H and a TB code $\mathcal{C}$ as described in Section 2.2.2.

Theorem 3. *Construction 3 yields an (n, k, r)-LRC $\mathcal{C}$ that is secure against an $(k-1, 0)$-eavesdropper.*

Proof: Since the inner storage code is an (n, k, r) TB code, the resulting code is an (n, k, r) LRC. To prove its security, assume without loss of generality that Eve observes $\ell = k - 1$ linearly independent symbols of $\mathcal{C}$. Let $E = \{i_1, \ldots, i_\ell\}$ be the indices of the symbols observed by Eve, *i.e.*, $C_E = G_E X$. Let $N = [N_1 \cdots N_k]^T \in \mathbb{F}_q^k$ be a vector orthogonal to G_E, *i.e.*, $G_E N = 0$. Due to the structure of G_E, we have that $\{\alpha_{i_1}, \cdots, \alpha_{i_\ell}\}$ are roots of the polynomial

$$N(x) = \sum_{j=1}^{k} N_j x^{[(j-1) \bmod r] + (r+1)\lfloor (j-1)/r \rfloor}.$$

Note that $N(x) \in \mathbb{F}_q[x]$, and it is easy to verify that the degree of $N(x)$ is at most $k + k/r - 2$. Thus, we can write $N(x) = \prod_{t=1}^{\ell}(x - \alpha_{i_t})N'(x)$, for some polynomial $N'(x) \in \mathbb{F}_q[x]$ with degree at most $k/r - 1$. Since β is a primitive element of $\mathbb{F}_{q^{k/r}}$, the degree of its *minimal polynomial* over $\mathbb{F}_q$ is k/r, and thus, β cannot be a root of $N'(x)$. Therefore, N cannot be in the null-space of H, *i.e.*, $HN \neq 0$. This guarantees that $\begin{bmatrix} H \\ G_E \end{bmatrix}$ has full row rank, ensuring the security by Lemma 2 (see (3.7)). ■

Remark 7. *Our idea in Construction 3 is to design the coset coding matrix H such that it has the same structure as G. Note that, if we append G with H, it can be considered as extending the TB code by adding an evaluation point β.*

Remark 8. *Since TB codes can be constructed over a field of size $q = O(n)$, Construction 3 gives secure LRCs over a field of size $Q = O\left(n^{k/r}\right)$. This demonstrates that the* *proposed construction is significantly better in terms of field size as compared to $O(n^k)$ when using Gabidulin precoding over an (n, k, r) TB code.*

Remark 9. *The authors of [42] bring out several connections between secure LRCs and secret sharing schemes that enable local recoverability for shares. In the perspective of secret sharing, Construction 3 presents a scheme that encodes a secret $S \in \mathbb{F}_Q$ into n shares over $\mathbb{F}_Q$ with the following properties. (i) Any $m = k + k/r - 1$ shares are sufficient to determine the secret S, which guaranties that the secret is recoverable even with the loss of any $n - m$ shares. (ii) Any $\ell = k - 1$ shares do not reveal any information about the secret. (iii) Any share can be locally recovered from at most r other shares.*

2.5 Conclusion

In this chapter, we have proposed a framework for design of MSR and MR codes that are secure against a passive eavesdropper. Our approach is to design a concatenated code that leverages existing non-secure storage code constructions to construct both inner and outer codes. The key idea is to utilize the structure of the generator matrix of a parent storage code to construct the parity check matrix of the outer coset code. This enabled us to reduce the required field size compared to the standard approaches such as precoding via Gabidulin codes.

3. WEAK SECURITY IN DISTRIBUTED STORAGE*

3.1 Introduction

Due to their decentralized nature, DSS are susceptible to adversarial attacks by malicious users and/or network operators. In such attacks, an adversary may attempt to gain access to the valuable information stored on the system or may want to modify the stored information in order to disrupt the system. Here, we focus on passive eavesdroppers that can observe the data stored on limited number of storage nodes.

We are interested in information-theoretic security models. Information-theoretic *perfect secrecy* requires that the eavesdropper gains absolutely *no information* about the stored data from its observations. To be precise, suppose that a DSS is storing B_s data files $S = \{S_1, \ldots, S_{B_s}\}$, where each file can be considered as a symbol in a finite field $\mathbb{F}_q$. Let E denote the set of (encoded) files that an eavesdropper *Eve* can observe. A DSS is said to be perfectly secure if the mutual information between the set of message symbols S and the eavesdropped symbols E is zero, *i.e.*, $I\left(S; E\right) = 0$.

For many practical storage systems, perfect secrecy condition might be too strong. Moreover, coding schemes that provide perfect secrecy involve mixing data symbols with random keys to confuse the eavesdropper, which incurs loss in the storage capacity. Considering these drawbacks of the perfect secrecy notion, we focus on the notion of *weak security* proposed by Bhattad and Narayanan [35].

Based on the premise that individual files carry meaningful information, the notion of weak security requires that Eve gains no information about any individual file, *i.e.*, $I\left(S_i; E\right) = 0 \ \forall i$. For example, let the number of files be $B_s = 4$, and the size of the

*Parts of this chapter are reprinted with permission from [50] “Weakly secure regenerating codes for distributed storage,” by S. Kadhe and A. Sprintson, 2014. *In Proceedings of 2014 International Symposium on Network Coding (NetCod)*, pp. 1-6, June 2014. Copyright © by IEEE.

underlying finite field be $q = 7$. Further, suppose that Eve observes the following two encoded symbols $E = \{S_1 + S_2 + S_3 + S_4,\ S_1 + 2S_2 + 3S_3 + 4S_4\}$. Then, Eve cannot get any information about any individual file, when the files are chosen uniformly at random and are independent. In other words, weakly secure coding schemes use files as keys, and thus, do not incur loss in storage capacity.

In addition, weak security requires that Eve cannot gain any information about *any* small group of files of size $(g + 1)$ for some positive integer g. In this case, even if Eve obtains any g files as a side information, she cannot decode for any other file. For instance, in the example considered above, even if Eve has a side-information of any $g = 1$ file, she cannot decode for any other file observing E.

Note that the notion of weak security that is introduced in [35] and considered throughout this paper, is different from the conventional notion of information-theoretic *weak secrecy*, which is defined for asymptotically large block-lengths. The weak security notion considered in this paper is applicable to finite block-lengths as well. The notion of weak security has also been referred to as *block security*, as it requires protecting blocks of information of different sizes (see, *e.g.* [37]).

In this chapter, we first consider a sub-class of regenerating codes, namely Product-Matrix (PM) codes in [13]. We present explicit construction of a coset coding based outer code to weakly secure PM codes against an eavesdropper observing any single storage node. Next, we construct a universal outer coding scheme that can weakly secure any systematic regenerating code operating at minimum storage point.

3.2 Preliminaries

3.2.1 Regenerating Codes

Consider a DSS that stores a set of B files given as $S = \{S_1, \dots, S_B\}$, where each file can be considered as a uniformly and independently drawn symbol from a finite field

$\mathbb{F}_q$. The system contains n storage nodes, with each node capable of storing α files. An (n, k, d, α, β) regenerating code encodes the B files into $n\alpha$ files in such a way that it satisfies the following two properties. (i) *Reconstruction property* – a *data collector* (DC) connecting to *any* k out of n nodes can reconstruct the entire set of files; (ii) *regeneration property* – when a storage node fails, it can be *regenerated* by adding a new node which downloads β symbols each from any d out of the remaining $n - 1$ nodes. The d nodes participating in node repair are referred to as the *helper nodes*, and the $d\beta$ number of symbols downloaded is referred to as the *repair bandwidth*.

Using the cut-set bounds, an upper bound on the capacity of an (n, k, d, α, β) regenerating code can be shown as [8]

$$B \leq \sum_{i=0}^{k-1} \min\{\alpha, (d-i)\beta\}. \tag{3.1}$$

It can be seen that, if one wants to achieve equality in (3.1) for fixed values of $n, k, d,$ and B, it leads to a trade-off between storage per node α and repair bandwidth $d\beta$. The two extreme points on this trade-off curve are referred to as minimum storage regeneration (MSR) point and minimum bandwidth regeneration (MBR) point. At the MSR point, which is obtained by first minimizing α and then β, we have

$$\alpha_{MSR} = \frac{B}{k}, \qquad \beta_{MSR} = \frac{B}{k(d-k+1)}. \tag{3.2}$$

Whereas, at the MBR point, for which first β is minimized and then α, we get

$$\alpha_{MBR} = \frac{2dB}{k(2d-k+1)}, \qquad \beta_{MBR} = \frac{2B}{k(2d-k+1)}. \tag{3.3}$$

If the regenerated node is an exact replica of the failed node, then the repair model is said to be *exact repair* [12]. In the first half of this chapter, we focus on a special family of exact regenerating codes, namely, the product-matrix (PM) codes [13], which are described in the next section.

3.2.2 Information-Theoretic Secrecy

Eavesdropper Model: We assume that an eavesdropper *Eve* can access the data stored in any l $(< k)$ storage nodes. Further, we assume that Eve is passive, has unbounded computational power, and has the knowledge of the coding scheme being used. For the first part of this chapter, we focus on the simple case of $l = 1$.

It is worth pointing out that, for an MSR code, the number of downloaded symbols $(d\beta)$ is strictly greater than the number of stored symbols (α). Therefore, Eve can potentially gain more information by observing the data downloaded during node repair than merely observing the data stored on the node. This motivates a generalized eavesdropper model for a DSS, called as the (l_1, l_2)-eavesdropper model, where, Eve can access the data stored on any l_1 nodes, and the data downloaded during the regeneration of any l_2 nodes (see [33, 22]). Our focus is on the case $l_1 = \ell$, $l_2 = 0$.

Perfect Seccurity: Suppose that we need to store a set S of B_s files *securely*, where $B_s \leq B$. Let E denote the set of (coded) files observed by Eve. A DSS is said to be *perfectly secure* if $I\left(S; E\right) = 0$. Under this requirement, Pawar *et al.* [31] characterized an upper bound on the *secrecy capacity* as:

$$B_s \leq \sum_{i=l}^{k-1} \min\{\alpha, (d-i)\beta\}, \tag{3.4}$$

where l is the number of nodes observed by Eve. Comparing (3.1) and (3.4), we can say that in a perfectly secure DSS, the l nodes that are compromised by the eavesdropper cannot effectively contain any useful information. Consequently, the perfect secrecy requirement results in a loss of storage capacity, *i.e.*, $B_s < B$.

For instance, in [33], the authors present a perfectly secure construction for product matrix codes that achieves the above outer bound. The loss of capacity incurred due to perfect secrecy requirement is $B - B_s = \alpha l$.

Weak Security: Our focus is on a relaxed, yet practically appealing notion of *weak security* [35]. A DSS is said to be *weakly secure* if $I\left(S_i; E\right) = 0$, $\forall i \in [B_s]$, where $[B_s] := \{1, \ldots, B_s\}$. Furthermore, suppose Eve is able to obtain, as a side information, some g files denoted as $S_{\mathcal{G}} := \{S_i : i \in \mathcal{G}\}$ for some $\mathcal{G} \subset [B_s]$ such that $|\mathcal{G}| = g$. Then, a DSS is said to be *weakly secure against g guesses* if we have

$$I\left(S_i; E | S_{\mathcal{G}}\right) = 0 \quad \forall i \in [B_s] \setminus \mathcal{G},\, \forall \mathcal{G} \subset [B_s] : |\mathcal{G}| \leq g. \tag{3.5}$$

Notice that when the system is weakly secure against g guesses, it means that the size of the side-information that Eve can possess without being able to decode any new file is (at most) g.

In [47], it was shown that the above condition is equivalent to

$$I\left(S_{\mathcal{G}'}; E\right) = 0 \quad \forall \mathcal{G}' \subseteq [B_s] : |\mathcal{G}'| \leq g + 1. \tag{3.6}$$

Essentially, this condition implies that in a scheme that is weakly secure against g guesses, it is not possible for Eve to obtain any information about *any* subset of $g+1$ files from her observations. We can consider the condition in (3.6) as the definition of weak security.

Coset-Coding-Based Outer Codes: A coset code [45] is constructed using a $(B, B - B_s)$ linear code $\mathcal{C}_s$ over $\mathbb{F}_q$ with parity-check matrix $H \in \mathbb{F}_q^{B_s \times B}$. Specifically, the set of B_s files S is encoded by selecting uniformly at random some $X \in \mathbb{F}_q^B$ such that $S = HX$. Therefore, the vector S can be considered as a syndrome specifying a coset of $\mathcal{C}_s$, and the codeword X is a randomly chosen element of that coset. Notice that the decoding operation of a coset code consists of simply computing the syndrome $S = HX$.

To design the matrix H appropriately, we need to evaluate the weak security condition (3.6) in terms of coding matrices H and G_e. For this, we consider the following result from [47, Lemma 6]. (See also [51, Theorem 1].)

Lemma 2. *Suppose a coset code based on a parity-check matrix $H \in \mathbb{F}_q^{B_s \times B}$ is used as an outer code over a given exact regenerating code to store the message $S = \{S_1, \cdots, S_{B_s}\}$. Suppose each message symbol S_i for $i \in [B_s]$ is chosen uniformly and independently. Let $E = GX$ be the μ linearly independent symbols observed by an eavesdropper. Then, for any $\mathcal{G}' \subseteq [B_s] : |\mathcal{G}'| \leq B - \mu$, we have*

$$I\left(S_{\mathcal{G}'}; E\right) = rank\, H_{\mathcal{G}'} + rank\, G - rank \begin{bmatrix} H_{\mathcal{G}'} \\ G \end{bmatrix}, \tag{3.7}$$

where $H_{\mathcal{G}'}$ is a sub-matrix of H formed by choosing the rows indexed by the set $\mathcal{G}'$.

Then, using (3.6) and (3.7), we can observe that the weak security is satisfied if H and G are such that

$$\text{rank} \begin{bmatrix} H_{\mathcal{G}'} \\ G \end{bmatrix} = \text{rank}\, H_{\mathcal{G}'} + \text{ rank}\, G_e, \;\; \forall\, |\mathcal{G}'| \leq g + 1. \tag{3.8}$$

Our goal is to construct the outer code H for Product Matrix codes (described in the next section) such that (3.30) holds.

Remark 10. *It is possible to choose entries of H uniformly at random over a sufficiently large finite field, however it would require exponential field size. This is because the condition* (3.30) *must be satisfied for all sub-matrices of H of size $(g + 1)$, the number of which is exponentially large. More specifically, using the standard network coding arguments, one can show that if entries of H are chosen uniformly at random from a field of* *size $q > n\binom{B}{B-\alpha}$, the scheme would be weakly secure against $B - \alpha$ number of guesses with high probability. On the other hand, we explicitly construct H over small field size $(\mathcal{O}(n))$ that guarantees weak security up to a number of guesses.*

3.3 Explicit Outer Codes for Weak Security of Product-Matrix (PM) Codes

3.3.1 Recap of PM Codes

The product-matrix code C is constructed as the product of an $(n \times d)$ encoding matrix Ψ and a $(d \times \alpha)$ message matrix M. The encoding matrix is chosen in such a way that it satisfies certain properties and is independent of message symbols. The message matrix contains the message symbols arranged in a particular fashion with possible repetitions. The constructions are specified for $\beta = 1$, and can be generalized for higher values of β by employing stripping of data (see [13]).

The PM-MBR code construction is specified for all feasible values of (n, k, d). The encoding matrix Ψ and message matrix M have the following structure

$$\underbrace{\Psi}_{n\times d} = \begin{bmatrix} \underbrace{\Phi}_{n\times k} & \underbrace{\Delta}_{n\times(d-k)} \end{bmatrix}, \quad \underbrace{M}_{d\times d} = \begin{bmatrix} \underbrace{M_1}_{k\times k} & \underbrace{M_2}_{k\times(d-k)} \\ \underbrace{M_2^T}_{(d-k)\times k} & \underbrace{0}_{(d-k)\times(d-k)} \end{bmatrix} \tag{3.9}$$

In the message matrix M, the component matrix M_1 is a $k \times k$ symmetric matrix which contains $\frac{k(k+1)}{2}$ data symbols in the upper triangular half; whereas, the other component matrix M_2 is a $k \times (d-k)$ matrix which contains the remaining $k(d-k)$ message symbols. Note that at MBR point, $B = \sum_{i=0}^{k-1}(d-i) = \frac{k(k+1)}{2} + k(d-k)$ (see (3.1)).

The matrices Φ and Δ are chosen in such a way that any k rows of Φ are linearly independent, and any d rows of Ψ are linearly independent. If Ψ is chosen to be a Vandermonde or a Cauchy matrix, these requirements are satisfied. Note that the field size q depends on the choice of the encoding matrix Ψ. For instance, if Ψ is a Vandermonde matrix, then field size of $q \geq n$ is necessary.

The α symbols stored on the i-th node are given by $C_i = \Psi_i M$, where Ψ_i denotes the i-th row of Ψ. The regeneration and the reconstruction processes can be found in [13].

Example 1. *Consider a* $(n = 5, k = 3, d = 4, \alpha = 4, \beta = 1)$ *PM-MBR code. Then, from* (3.3)*, we have* $B = 9$*. Let the data to be stored is given as* $X = \{x_1, \cdots, x_9\}$*, where* $x_i \in \mathbb{F}_q\ \forall i$*. Suppose the encoding matrix* Ψ *is a Cauchy matrix. Then, in parametric form, we have*

$$\Psi = \left[\frac{1}{a_i + b_j}\right]_{i=1,j=1}^{5,4}, \quad M = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_5 & x_6 & x_7 \\ x_3 & x_6 & x_8 & x_9 \\ x_4 & x_7 & x_9 & 0 \end{bmatrix}, \tag{3.10}$$

where $a_i, b_j \in \mathbb{F}_q$ *such that* $a_i \neq b_j$ *and* $a_i + b_j \neq 0$ *for all* i, j*. Note that, to satisfy these requirements, we need at least* $n + d = 9$ *distinct elements, and thus, we require* $q \geq 9$*.*

3.3.2 Conventional Representation of PM Codes

Let us begin with describing a conventional way to represent the PM codes in terms of a generator matrix and a message vector. This will be useful later in specifying the outer codes.

Let C_e denote the α files stored on a node e. Then, recall that $C_e = \psi_e M$, where ψ_e is the e-th row of Ψ. We can represent this encoding vector times message matrix form in a more conventional generator matrix times message vector form. This is carried out by finding an $\alpha \times B$ matrix G_e that guarantees $C_e = \psi_e M = (G_e X)^T$, where X is a column vector containing all the B files.

Example: For the previous example of $(n = 6, k = 4, d = 5, \alpha = 5, \beta = 1)$ PM-MBR code (Example 1), if Eve observes the first node then we can write G_1 as (3.11) as follows.

$$G_1 = \begin{bmatrix} \Psi(1,1) & \Psi(1,2) & \Psi(1,3) & \Psi(1,4) & 0 & 0 & 0 & 0 & 0 \\ 0 & \Psi(1,1) & 0 & 0 & \Psi(1,2) & \Psi(1,3) & \Psi(1,4) & 0 & 0 \\ 0 & 0 & \Psi(1,1) & 0 & 0 & \Psi(1,2) & 0 & \Psi(1,3) & \Psi(1,4) \\ 0 & 0 & 0 & \Psi(1,1) & 0 & 0 & \Psi(1,2) & 0 & \Psi(1,3) \end{bmatrix} \tag{3.11}$$

Remark 11. *Observe that matrix G_e for each node $e \in [n]$ is sparse. In particular, G_e for each node $e \in [n]$ contains at least one row vector with Hamming weight k. Thus, PM-MBR codes are not secure against $g \geq k-1$ guesses, when Eve can observe one storage node. This shows the necessity to employ an outer code to improve the level of weak secrecy.*

3.3.3 Main Idea and Definitions

The idea is to construct H such that it has the same *structure* as that of the generator matrix G_e of a node for the PM code. The same structure of G_e and H would enable us to ensure the condition (3.30).

Recall from section 3.3.2 that the values of the non-zero entries in G_e are specified by the elements of ψ_e and their locations depend on the indices of the elements of M. Further, the locations of non-zero entries in G_e are the same for each node $e \in [n]$. To formally specify this *structure* present in G_e, we first introduce the notion of *type*.

Definition 3. *A row vector h_j, of length B is said to be of type i, $1 \leq i \leq \alpha$, if the location of the non-zero entries of h_j are the same as that of i-th row of G_e, where G_e is the generator matrix of the underlying PM code. We call the corresponding set of indices of non-zero entries as the index set of type i, denoted as $\mathcal{I}(i)$.*

Essentially, the type of a vector specifies the locations of the non-zero entries of the vector. Notice that the total number of types is equal to the number of rows of G_e, which is α. Note that the underlying PM code to be secured determines the number of types and structure of each type.

Example 2. *Continuing the example of an* $(n = 6, k = 4, d = 5, \alpha = 5, \beta = 1)$ *PM-MBR code (Example 1), a vector of type 1 has the form*

$$h^{(1)} = \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

for any non-zero elements $\{\gamma_1, \cdots, \gamma_4\}$ *in* $\mathbb{F}_q$*. Note that its structure resembles that of the first row of* G_1 *in* (3.11)*. The corresponding index set of type 1 is* $\mathcal{I}(1) = \{1, 2, 3, 4\}$*, which corresponds to the indices of elements of first column of* M *(see* (3.10)*).*

As we show in the next section, for a given PM code, we propose to construct H such that each of its rows corresponds to one of the α types. This motivates us to introduce the notion of *type cardinality vector*.

Definition 4. *For a* $B_s \times B$ *matrix* H*, we define its type cardinality vector* θ *as* $\theta := [\theta_1 \cdots \theta_\alpha]$*, where* θ_i *denotes the number of rows of type* i*,* $1 \leq i \leq \alpha$*, that are present in* H*.*

Once the type of a row vector is fixed, it is sufficient to give the set of values of its non-zero entries to fully specify the row vector. For example, the values of non-zero entries of all the vectors in G_1 in (3.11) are specified by the elements of vector $\psi_2 = [\Psi(1,1) \;\; \Psi(1,2) \;\; \Psi(1,3) \;\; \Psi(1,4)]$. If all the rows of H are of one of the α types, we can succinctly represent all the non-zero entries H using what we call as a *coefficient matrix* defined below.

Definition 5. *For a* $B_s \times B$ *matrix* H*, its coefficient matrix* $\hat{\Psi}$ *is defined as the* $d \times d$ *matrix* *such that the j-th row of $\hat{\Psi}$ specifies the non-zero entries of the j-th vector of type i that is present in H for $i \in [\alpha], j \in [d]$.*

Observe that the type cardinality vector θ along with the coefficient matrix $\hat{\Psi}$ completely specify the parity-check matrix H.

Example 3. *Consider the example of an $(n = 6, k = 4, d = 5, \alpha = 5, \beta = 1)$ PM-MBR code . Notice that we have $\alpha = 5$ number of types, whose structures are given by the rows of a generator matrix of a node, say G_1 (see (3.11)). Consider the H matrix as follows.*

$$H = \begin{bmatrix} 15 & 2 & 12 & 5 & 0 & 0 & 0 & 0 & 0 \\ 2 & 12 & 14 & 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 0 & 0 & 2 & 12 & 5 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 & 0 & 2 & 0 & 12 \\ 0 & 0 & 0 & 2 & 0 & 0 & 12 & 0 & 14 \end{bmatrix}. \tag{3.12}$$

Then, we can represent H using the following type cardinality vector and coefficient matrix.

$$\theta = \{2, 1, 0, 2\}, \quad \hat{\Psi} = \begin{bmatrix} 15 & 2 & 12 & 5 \\ 2 & 12 & 14 & 8 \end{bmatrix}. \tag{3.13}$$

3.3.4 Outer Code Construction for PM-MBR Codes

For PM-MBR codes, an explicit joint construction of H and Ψ is given below.

Construction 4. *First, choose the type cardinality vector θ as follows.*

$$\theta_i = \begin{cases} 0 & \text{if} \quad i = 1, \\ d - k + i & \text{if} \quad 2 \leq i \leq k - 1, \\ d - 1 & \text{if} \quad i = k, \\ 1 & \text{if} \quad k + 1 \leq i \leq d. \end{cases} \tag{3.14}$$

Note that $\max_{1 \leq i \leq d} \theta_i = d - 1$.

Next, choose an $n \times d$ encoding matrix Ψ and a $d \times d$ coefficient matrix $\hat{\Psi}$ in such a way that $\tilde{\Psi} := \begin{bmatrix} \Psi \\ \hat{\Psi} \end{bmatrix}$ is a Cauchy matrix.

Finally, using $\hat{\Psi}$ and θ, construct H as follows. For each θ_i, $2 \leq i \leq d$, the $\theta_i \times B$ sub-matrix H_i of H is given as

$$H_i(p, b) = \begin{cases} \hat{\Psi}_{(p,j)} & \text{if } M_{(i,j)} = X_b \text{ for some } j \in [d], \\ 0 & \text{otherwise}, \end{cases} \tag{3.15}$$

for $1 \leq p \leq \theta_i$ and $1 \leq b \leq B$. The parity-check matrix H is obtained by vertically concatenating the sub-matrices H_i, i.e., $H = \begin{bmatrix} H_2^T \; H_3^T \; \ldots \; H_d^T \end{bmatrix}^T$. Notice that H_i is the sub-matrix of H formed by choosing all its rows corresponding to type i.

To construct $\tilde{\Psi}$ to be a Cauchy matrix, it suffices to have $q \geq n + 2\alpha$.

Example 4. *Consider the example of an $(n = 5, k = 3, d = 4, \alpha = 4, \beta = 1)$ PM-MBR code (Example 1). Construction 4 yields $\theta = \{0, 3, 3, 1\}$. Let the 5×4 encoding matrix Ψ be a Cauchy matrix (cf. (3.10)). We choose $\hat{\Psi}$ such that $\hat{\Psi} = \begin{bmatrix} \Psi \\ \hat{\Psi} \end{bmatrix}$ is also a Cauchy matrix. Note that this requires $q \geq 13$. Then, the resulting parity-check matrix H is given* *in* (3.16).

$$H = \begin{bmatrix} 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & \hat{\Psi}(1,3) & \hat{\Psi}(1,4) & 0 & 0 \\ 0 & \hat{\Psi}(2,1) & 0 & 0 & \hat{\Psi}(2,2) & \hat{\Psi}(2,3) & \hat{\Psi}(2,4) & 0 & 0 \\ 0 & \hat{\Psi}(3,1) & 0 & 0 & \hat{\Psi}(3,2) & \hat{\Psi}(3,3) & \hat{\Psi}(3,4) & 0 & 0 \\ 0 & 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & 0 & \hat{\Psi}(1,3) & \hat{\Psi}(1,4) \\ 0 & 0 & \hat{\Psi}(2,1) & 0 & 0 & \hat{\Psi}(2,2) & 0 & \hat{\Psi}(2,3) & \hat{\Psi}(2,4) \\ 0 & 0 & \hat{\Psi}(3,1) & 0 & 0 & \hat{\Psi}(3,2) & 0 & \hat{\Psi}(3,3) & \hat{\Psi}(3,4) \\ 0 & 0 & 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & 0 & \hat{\Psi}(1,3) \end{bmatrix}. \tag{3.16}$$

First, we characterize the weak security capacity of the proposed outer code for PM-MBR codes.

Theorem 4. *When an outer coset code based on the parity-check matrix H given in Construction 4 is used along with a PM-MBR code, the weakly secure storage capacity is $B_s = B - 2$.*

Proof: Notice that the data file, which is securely stored, can be considered as the syndrome of the coset code as $S = HX$. Thus, the weak-secrecy capacity is the dimension of matrix H. First, we show that, if H is designed following construction 4, it contains $B - 2$ rows. Next, we show that H is full-rank to prove the result.

Now, notice that the total number of rows in H is equal to $\sum_{i=1}^{d} \theta_i$. From (3.14), we have

$$\begin{aligned}\sum_{i=1}^{d}\theta_i &= 0+\left(\sum_{i=2}^{k-1} d-k+i\right)+(d-1)+(d-k)\\ &= \left(\sum_{i=1}^{k-2} d-i\right)+(d-1)+(d-k)\\ &\overset{(a)}{=} \left(\sum_{i=0}^{k-1} d-i\right)-2\\ &\overset{(b)}{=} B-2 \end{aligned} \tag{3.17}$$

where (a) can be easily obtained by carrying out simple algebraic manipulations, and (b) follows from (3.1) and from the fact that at MBR point $\alpha = d$ (we assume that $\beta = 1$).

To prove that H is full-rank, we show that it is possible to append two rows to H in such a way that the resulting $B \times B$ matrix, denoted as H', is non-singular. Specifically, append a type 1 row vector with non-zero coefficients corresponding to the first row of $\hat{\Psi}$, and append a type k row vector with non-zero coefficients corresponding to the d-th row of $\hat{\Psi}$. From (3.14), it is easy to see that H' contains $d-(k-i)$ rows of type i for $k \geq i \geq 2$, one row of type 1, and one row each of types $k+1$ through d. Without loss of generality, assume that the rows of H are ordered in such a way that first d rows are of type k, next $d-1$ rows are of type $k-1$ and so on up to $d-(k-2)$ rows of type 2. The last $d-k+1$ rows are of types $k+1$ through d and of type 1, respectively.

Now, for proving non-singularity of H', consider a system of linear equations $Z = H'Y$, where $Y = [Y_1 \cdots Y_B]$ and $Z = [Z_1 \cdots Z_B]$ are length-B vectors of variables Y_1 through Y_B and Z_1 through Z_B, respectively. We show that it is possible to *successively decode* variables in Y in terms of variables in Z by leveraging the elegant structure of H'.

To describe the process of successive decoding, we need to introduce some notation. Recall that the type of a row vector specifies the locations of the non-zero coefficients of

named 1 Successive decoding for $Z = H'Y$

```
Set $\gamma_1 = d$, $\gamma_2 = k$
for $p = k$ to 2 do
    Consider set of equations corresponding to $\gamma_1$ rows of type $p$ as $Z[\mathcal{I}(p)] = \hat{\Psi}_{1:\gamma_1} Y[\mathcal{I}(p)]$
    Using perviously decoded elements, decode for elements of $Y$ located at $\mathcal{I}(p) \setminus \left(\bigcup_{l=1}^{k-p} (\mathcal{I}(p) \cap \mathcal{I}(p+l))\right)$
    $\gamma_1 = \gamma_1 - 1$, $\gamma_2 = \gamma_2 - 1$
end for
Decode for the remaining elements in index sets of types $k+1$ through $d$
Decode for the remaining single element of type $1$
```

the vector. The corresponding set of locations of non-zero coefficients of a type i vector is referred to as the index set of type i, denoted as $\mathcal{I}(i)$. Define $Y[\mathcal{I}(i)] := \{Y_l : l \in \mathcal{I}(i)\}$, *i.e.*, $Y[\mathcal{I}(i)]$ is the vector of elements of Y that are indexed by the index set of type i.

Suppose we write vector Y in the format of matrix M (see (3.9)) in a lexicographic order. Notice that the index set of type i is the set of indices of the elements in the i-th column of M. Observing the structure of M, we divide the d types into two groups. The types 1 through k are called as group I, while the types $k+1$ through d are called as group II. For any group I type, the index set consists of d elements, *i.e.*, $|\mathcal{I}(i)| = d$, $\forall i \in [k]$. On the other hand, for any group II type, the corresponding index set has k elements, *i.e.*, $|\mathcal{I}(i)| = k$ $\forall i \in \{k+1, \ldots, d\}$. Further, index set corresponding to any group I has one index common with the index sets of all other types, *i.e.*, $|\mathcal{I}(i) \cap \mathcal{I}(j)| = 1$ $\forall i < j : i \in [k]$. Whereas, any pair of index sets of group II types has no common symbol, *i.e.*, $|\mathcal{I}(i) \cap \mathcal{I}(j)| = 0$ $\forall k < i < j \leq d$.

Let γ_1 and γ_2 denote the number of row vectors in H' of group I and group II types, respectively. Algorithm 1 presented below decodes elements of $Y[\mathcal{I}(i)]$ for each $i \in [d]$ successively.

Claim 1. *Algorithm 1 decodes all the B elements of Y in terms of Z.*

Proof: The algorithm begins with type k, of which there are d row vectors in H'. By the construction of H', the non-zero coefficient values are the elements of the $d \times d$ Cauchy matrix $\hat{\Psi}$. Thus, it is possible to solve for $Y[\mathcal{I}(k)]$ by inverting $\hat{\Psi}$. Next, we prove by induction that, for $2 \leq i \leq k$, if the elements of $\mathcal{I}(i+1)$ through $\mathcal{I}(k)$ have been decoded, then it is possible to decode the elements of $\mathcal{I}(i)$. By construction of H', there are $d-k+i$ rows of type i in H' for $2 \leq i \leq k$ with non-zero coefficients given by $\hat{\Psi}_{1:(d-k+i)}$, respectively. This forms a system of $d-k+i$ linear equations in d variables of $\mathcal{I}(i)$ as $Z[\mathcal{I}(i)] = \hat{\Psi}_{1:(d-k+i)}Y[\mathcal{I}(i)]$. Note that, since type i is a group I type, there is one element common between $\mathcal{I}(i)$ and each of $\mathcal{I}(i+1)$ through $\mathcal{I}(k)$. Thus, there are $k-i$ elements in $\mathcal{I}(i)$ that have already been decoded. As any square sub-matrix of $\hat{\Psi}$ is non-singular by construction (it is a Cauchy matrix), it is possible to solve for the un-decoded $d-k+i$ variables using $Z[\mathcal{I}(i)] = \hat{\Psi}_{1:(d-k+i)}Y[\mathcal{I}(i)]$.

At the end of the first **for** loop, $k-1$ elements from $\mathcal{I}(j)$ for each $j \in [d]$ are decoded. Thus, in each of the index sets of group II types, there is just one element to be decoded. By construction, H' has one row in each of the group II types, and thus, it is possible to decode all the elements in group II index sets. Note that, at this point, all the elements from index sets of all types except type 1 are decoded.

Finally, since $\mathcal{I}(1)$ has one element common with all the remaining $d-1$ index sets, only single element from $\mathcal{I}(1)$ remains to be decoded, which can be decoded using a row of type 1 that is appended to H.

Notice that, as each index set corresponds to a column of matrix M, we have $\bigcup_{j=1}^{d} \mathcal{I}(j) = \{Y_1, \ldots, Y_B\}$. Therefore, algorithm 1 decodes all the B elements of Y. ■

From Claim 1, it follows that the matrix H' is invertible, which completes the proof.

■

Remark 12. *Note that successively decoding for the variables of a particular type is equiv-*

alent to performing Gaussian elimination on the corresponding rows of that particular type. Thus, in essence, the procedure for successive decoding gives the order in which Gaussian elimination can be performed in H'.

Example: Consider the following H given in (3.18), which follows from construction 4. To form H', first append a row vector of type 3 with non-zero coefficients specified by $\hat{\Psi}_4$. Then, append a row vector of type 1 with non-zero coefficients specified by $\hat{\Psi}_1$. See equation (3.19). To prove that H' is full-rank, observe that we can decode for variables indexed by $\mathcal{I}\left(3\right) = \{2, 5, 6, 7\}$ using the four rows of type 3. Notice that $\mathcal{I}\left(3\right) \cap \mathcal{I}\left(2\right) = 6$. Then, using the three rows of type 2 and already decoded variable at index 6, solve for variables indexed by $\mathcal{I}\left(2\right) \setminus \left(\mathcal{I}\left(3\right) \cap \mathcal{I}\left(2\right)\right) = \{3, 8, 9\}$. Then, using the row vector of type 4, decode for $\mathcal{I}\left(4\right) \setminus \left(\left(\mathcal{I}\left(4\right) \cap \mathcal{I}\left(3\right)\right) \cup \left(\mathcal{I}\left(4\right) \cap \mathcal{I}\left(2\right)\right)\right) = \{4\}$. Finally, using the row of type one, decode for $\mathcal{I}\left(1\right) \setminus \left(\left(\mathcal{I}\left(1\right) \cap \mathcal{I}\left(4\right)\right) \cup \left(\mathcal{I}\left(1\right) \cap \mathcal{I}\left(3\right)\right) \cup \left(\mathcal{I}\left(1\right) \cap \mathcal{I}\left(2\right)\right)\right) = \{1\}$. The successive decoding uses the property that any square sub-matrix of the Cauchy matrix $\hat{\Psi}$ is non-singular.

$$H = \begin{bmatrix} 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & \hat{\Psi}(1,3) & \hat{\Psi}(1,4) & 0 & 0 \\ 0 & \hat{\Psi}(2,1) & 0 & 0 & \hat{\Psi}(2,2) & \hat{\Psi}(2,3) & \hat{\Psi}(2,4) & 0 & 0 \\ 0 & \hat{\Psi}(3,1) & 0 & 0 & \hat{\Psi}(3,2) & \hat{\Psi}(3,3) & \hat{\Psi}(3,4) & 0 & 0 \\ 0 & 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & 0 & \hat{\Psi}(1,3) & \hat{\Psi}(1,4) \\ 0 & 0 & \hat{\Psi}(2,1) & 0 & 0 & \hat{\Psi}(2,2) & 0 & \hat{\Psi}(2,3) & \hat{\Psi}(2,4) \\ 0 & 0 & \hat{\Psi}(3,1) & 0 & 0 & \hat{\Psi}(3,2) & 0 & \hat{\Psi}(3,3) & \hat{\Psi}(3,4) \\ 0 & 0 & 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & 0 & \hat{\Psi}(1,3) \end{bmatrix} \tag{3.18}$$

$$H' = \begin{bmatrix} 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & \hat{\Psi}(1,3) & \hat{\Psi}(1,4) & 0 & 0 \\ 0 & \hat{\Psi}(2,1) & 0 & 0 & \hat{\Psi}(2,2) & \hat{\Psi}(2,3) & \hat{\Psi}(2,4) & 0 & 0 \\ 0 & \hat{\Psi}(3,1) & 0 & 0 & \hat{\Psi}(3,2) & \hat{\Psi}(3,3) & \hat{\Psi}(3,4) & 0 & 0 \\ 0 & 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & 0 & \hat{\Psi}(1,3) & \hat{\Psi}(1,4) \\ 0 & 0 & \hat{\Psi}(2,1) & 0 & 0 & \hat{\Psi}(2,2) & 0 & \hat{\Psi}(2,3) & \hat{\Psi}(2,4) \\ 0 & 0 & \hat{\Psi}(3,1) & 0 & 0 & \hat{\Psi}(3,2) & 0 & \hat{\Psi}(3,3) & \hat{\Psi}(3,4) \\ 0 & 0 & \hat{\Psi}(4,1) & 0 & 0 & \hat{\Psi}(4,2) & 0 & \hat{\Psi}(4,3) & \hat{\Psi}(4,4) \\ 0 & 0 & 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & 0 & \hat{\Psi}(1,3) \\ \hat{\Psi}(1,1) & \hat{\Psi}(1,2) & \hat{\Psi}(1,3) & \hat{\Psi}(1,4) & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.19}$$

Next, we compute the level of secrecy that can be attained using the proposed outer code along with a PM-MBR code.

Theorem 5. *An outer coset code based on the parity-check matrix H given in Construction 4 makes a PM-MBR code weakly secure against $g \leq d + k - 4$ guesses.*

Proof: Essentially, we need to prove that condition (3.30) always holds for the proposed coding scheme as long as $|\mathcal{G}'| \leq d + k - 3$. For notational convenience, let $T := \begin{bmatrix} H_{\mathcal{G}'} \\ G_e \end{bmatrix}$. Notice that there are $\binom{B_s}{|\mathcal{G}'|}$ number of ways to choose a particular $|\mathcal{G}'|$, and we ensure (3.30) for each possible $H_{\mathcal{G}'}$ as long as $|\mathcal{G}'| \leq d + k - 3$.

Since H is full-rank as shown in theorem 4, its sub-matrix $H_{\mathcal{G}'}$ will be full rank for any $\mathcal{G}' \subseteq [B_s]$. Further, it has been shown in [33] that for PM-MBR codes each storage node stores α linearly independent symbols, thus, it follows that G_e is full-rank. Therefore, to prove (3.30), we need to prove that the matrix T is full-rank. We divide the proof into three cases: $k \geq 3$, $k = 2$, and $k = 1$.

Case 1: $k \geq 3$.

As in the proof of Theorem 4, we show that, if $|\mathcal{G}'| \leq d + k - 3$, it is always possible to append $B - |\mathcal{G}'| - \alpha$ rows of appropriate types to T in such a way that the resulting $B \times B$ matrix is non-singular. In the following, we present an algorithm which, for any given $H_{\mathcal{G}'}$ and node index $e \in [n]$, appends row vectors of appropriate types to $T = \begin{bmatrix} H_{\mathcal{G}'} \\ G_e \end{bmatrix}$ in such a way that successive decoding can be carried out.

We need some notation to describe the algorithm. Let λ_i' be the number of encoding vectors of type i, $i \in [d]$, that are present in $H_{\mathcal{G}'}$. Notice that $\lambda_i' \leq \theta_i \; \forall i \in [d]$. Let λ_i denote the number of row vectors of type i, $i \in [d]$, that are present in T. Note that, for any $e \in [n]$, G_e contains one row vector of each of the d types. Thus, $\lambda_i = \lambda_i' + 1, \forall i \in [d]$. This implies that $\lambda_i \leq \theta_i + 1 \; \forall i \in [d]$. Further, from (3.14), we have that $\lambda_i \in \{1, 2\}$ for all group II types i for $k + 1 \leq i \leq d$. Let γ_1 and γ_2 denote the number of equations that are required to decode the variables of group I and group II types, respectively, in a given iteration. See algorithm 2 on next page.

First, we prove the correctness of the algorithm.

Claim 2. *If algorithm 2 does not report a failure, it finds a solution to $Z = T'Y$, where the construction of T' is described in the algorithm.*

Proof: In the same way as in the proof of claim 1, it is easy to prove by induction that in the first **for** loop, the algorithm decodes for $Y[\mathcal{I}(j_i)]$ for $3 \leq i \leq k$. Since each pair of group I types has one index in common, $k - 2$ elements of each of the remaining types are decoded at the end of the first **for** loop. Note that there are two rows each of types j_{d-L+1} through j_d. Since $k - 2$ elements of each of them are already decoded, the remaining two elements are decoded at line 17.

At line 25, all the remaining elements of $\mathcal{I}(j_2)$ will be decoded. At this point, there is only one un-decoded element each in types j_{k+1} through j_{d-L}, which will be decoded at line 28. Note that, at this point, all the types from 2 through d have been decoded. Thus, there remains only one un-decoded element of type 1 which will be decoded as the final

named 2 Appending rows to T to form T' and carrying out successive decoding for $Z = T'Y$ $(k \geq 3)$

```
Sort $\lambda_j$ for $j \in [k]$, Let $\lambda_{j_1} \leq \cdots \leq \lambda_{j_k}$
Sort $\lambda_j$ for $k+1 \leq j \leq d$, Let $\lambda_{j_{k+1}} \leq \cdots \leq \lambda_{j_d}$
Find $L$ such that $\lambda_{j_{d-L+1}} = \lambda_{j_{d-L+2}} = \cdots = \lambda_{j_d} = 2$
{Notice that $0 \leq L \leq d-k$}
Set $\gamma_1 = d$, $\gamma_2 = k$
for $p = k$ to $3$ do
    if $\lambda_{j_p} > \gamma_1$ then
        Declare failure, Exit
    else
        Append $T$ with $\gamma_1 - \lambda_{j_p}$ additional rows of type $j_p$ with non-zero coefficients as
        the rows of $\hat{\Psi}$ that are not present in the $\lambda_{j_p}$ rows of type $j_p$
        Using the equations corresponding to the $\gamma_1$ rows of type $p$, decode the un-
        decoded variables from $Y[\mathcal{I}(j_p)]$
        $\gamma_1 = \gamma_1 - 1$, $\gamma_2 = \gamma_2 - 1$
    end if
end for
{At this point, $\gamma_1 = d - (k-2)$ and $\gamma_2 = k - (k-2) = 2$}
if $L > 0$ then
    Successively decode the remaining variables from $Y[\mathcal{I}(j_{d-L+i})]$ for $i \in [L]$
    $\gamma_1 = \gamma_1 - L$
end if
{At this point, $\gamma_1 = d - (k-2) - L$ and $\gamma_2 = 2$}
if $\lambda_{j_2} > \gamma_1$ then
    Declare failure, exit
else
    Append $T$ with $\gamma_1 - \lambda_{j_2}$ additional rows of type $j_2$ with non-zero coefficients as the
    rows of $\hat{\Psi}$ that are not present in the $\lambda_{j_2}$ rows of type $j_2$
    Decode the un-decoded variables from $Y[\mathcal{I}(2)]$
    $\gamma_1 = \gamma_1 - 1$, $\gamma_2 = \gamma_2 - 1$
    {At this point, $\gamma_1 = d - (k-2) - L - 1$ and $\gamma_2 = 2 - 1 = 1$}
    Decode for the remaining symbols from $Y[\mathcal{I}(j_{k+1})], Y[\mathcal{I}(j_{k+2})], \ldots, Y[\mathcal{I}(j_{d-L})]$
    Append $T$ with a row of type $1$ with non-zero coefficients as the first row of $\hat{\Psi}$
    Decode the un-decoded variable from $Y[\mathcal{I}(1)]$
end if
```

step. For successive decoding, we rely on the fact that for matrix $\tilde{\Psi}_e = \begin{bmatrix} \hat{\Psi} \\ \Psi_e \end{bmatrix}$, any square sub-matrix is non-singular. Note that this condition holds by construction 1; e.g., when the matrix $\tilde{\Psi}$ is a Cauchy matrix.

Essentially, algorithm decodes the elements in all the d types in the following order (j_k, j_{k-1}, . . ., j_3), (j_{d-L+1}, . . ., j_d), (j_{k+1}, . . ., j_{d-L}), j_2, j_1, which covers all the B elements. ■

Next, we prove that the algorithm 2 does not declare a failure if the number rows in $H_{\mathcal{G}'}$ is bounded below a threshold.

Claim 3. *If $|\mathcal{G}'| \leq d + k - 3$, then algorithm 2 always succeeds.*

Proof: The proof is by contradiction. Suppose $|\mathcal{G}'| \leq d + k - 3$, and the algorithm fails.

Case 1: Algorithm fails on line 6, *i.e.*, in the first iteration when $p = k$. This implies that $\lambda_{j_k} > \gamma_1 = d$. However, we have

$$\begin{aligned} \lambda_{j_k} &= \max_{1 \leq l \leq k} \lambda_l \\ &\overset{(a)}{\leq} \max_{1 \leq l \leq k} \theta_l + 1 \\ &\overset{(b)}{=} d \end{aligned}$$

where (a) follows from the fact that the number of rows of any type i in T is at most $\theta_i + 1$, *i.e.*, $\lambda_i \leq \theta_i + 1 \; \forall i \in [d]$, and (b) is due to (3.14). This is a contradiction, and the algorithm cannot fail in the first iteration when $p = k$.

Case 2: Algorithm fails in the first **for** loop during i-th iteration such that $2 \leq i \leq k-2$. Note that this implies $k \geq 4$. Also, at the i-th iteration, we have $p = k - (i - 1)$.

Now, as γ_1 is reduced by 1 in each iteration, in i-th iteration we have $\gamma_1 = d - (i - 1)$.

Since the algorithm failed, it should be that $\lambda_{j_{k-i+1}} > d - (i-1)$. Then, we can write

$$\begin{aligned} \sum_{l=1}^{i} \lambda_{j_{k-l+1}} &\overset{(c)}{\geq} i\lambda_{j_{k-i+1}} \\ &\overset{(d)}{\geq} i(d-i+2) \\ \therefore \quad i + \sum_{l=1}^{i} \lambda'_{j_{k-l+1}} &\geq i(d-i+2) \qquad (3.20) \\ \therefore \quad \sum_{l=1}^{i} \lambda'_{j_{k-l+1}} &\geq i(d-i+1), \qquad (3.21) \end{aligned}$$

where (c) is due to $\lambda_{j_k} \geq \lambda_{j_{k-1}} \geq \cdots \geq \lambda_{j_{k-i+1}}$ and (d) is due to $\lambda_{j_{k-i+1}} > d-(i-1)$. To get (3.20), we use $\lambda_l = \lambda'_l + 1 \; \forall l \in [d]$.

First, note that $|\mathcal{G}'| = \sum_{l=1}^{d} \lambda'_l$. Thus, $|\mathcal{G}'| \geq \sum_{l=1}^{i} \lambda'_{j_{k-l+1}}$. Next, notice that $f(i) = i(d-i+2)$ is a concave function in i and thus it will attain minimum over $2 \leq i \leq k-2$ at one of its boundary points. Using these two observations along with (3.21), we have

$$|\mathcal{G}'| \geq \min\{2(d-1), (d-k+3)(k-2)\}. \qquad (3.22)$$

Now, we show that both of these boundary points result in contradiction. First, since $d \geq k$ for any regenerating code, clearly, $2(d-1) - (d+k-3) = d-k+1 > 0$, *i.e.*, $2(d-1) > d-k+3$. Next, consider the second boundary point,

$$\begin{aligned} (d-k+3)(k-2) - (d+k-3) &\overset{(e)}{=} (k-3)d - (k-2)^2 + 1 \\ &\overset{(f)}{\geq} (k-3)k - (k-2)^2 + 1 \\ &= k-3, \end{aligned}$$

where (e) follows from algebraic manipulations, and (f) follows because for any regenerating code $d \geq k$. Finally, since $k \geq 4$ in this case, we have $(d-k+2)(k-2) > d+k-3$.

Therefore, we have $|\mathcal{G}'| \geq \min\{2(d-1), (d-k+3)(k-2)\} > d+k-3$, which is a contradiction.

Case 3: Algorithm fails at line 22, while considering type j_2. It is easy to see that at this point $\gamma_1 = d-(k-2)-L$ and $\gamma_2 = 1$. The failure implies that $\lambda_{j_2} \geq d-(k-2)-L+1$. Now, let us consider the total number of rows in T corresponding to types that have been considered so far, as follows.

$$\begin{aligned} \sum_{l=2}^{k} \lambda_{j_l} + \sum_{m=d-L+1}^{d} \lambda_{j_m} &\overset{(g)}{\geq} \sum_{l=2}^{k} \lambda_{j_2} + 2L \\ &\overset{(h)}{\geq} (k-1)(d-k-L+3) + 2L, \end{aligned} \tag{3.23}$$

where (g) follows from $\lambda_{j_2} \leq \lambda_{j_3} \leq \cdots \leq \lambda_{j_k}$ and $\lambda_{j_{d-L+1}} = \lambda_{j_{d-L+2}} = \cdots = \lambda_{j_d} = 2$, and (h) follows from $\lambda_{j_2} \geq d-(k-2)-L+1$. However, since $\lambda_l = \lambda'_l + 1$ for each $l \in [d]$, from (3.23), we can write

$$\begin{aligned} \sum_{l=2}^{k} \lambda'_{j_l} + \sum_{m=d-L+1}^{d} \lambda'_{j_m} &\geq (k-1)(d-k-L+3) - (k-1) + L \\ &= (k-1)(d-k-L+2) + L. \end{aligned} \tag{3.24}$$

After some manipulations, it is straightforward to show that $(k-1)(d-k-L+2)+L-(d+k-3) = (d-k-L)(k-2)+1$, which is strictly positive for $k \geq 3$ as $d \geq k$ and $0 \leq L \leq d-k$. Therefore, we have

$$|\mathcal{G}'| = \sum_{l=1}^{d} \lambda'_{j_l} \geq \sum_{l=2}^{k} \lambda'_{j_l} + \sum_{m=d-L+1}^{d} \lambda'_{j_m} \overset{(o)}{\geq} (k-1)(d-k-L+2)+L \overset{(r)}{>} d+k-3, \tag{3.25}$$

where (o) follows from (3.24) and (r) is proved in the previous point. However, this is a contradiction, which completes the proof for $k \geq 3$.

■

named 3 Appending T and carrying out successive decoding for $k = 2$

```
Sort $\lambda_j$ for $k+1 \leq j \leq d$, Let $\lambda_{j_{k+1}} \leq \cdots \leq \lambda_{j_d}$
Find $L$ such that $\lambda_{j_{d-L+1}} = \lambda_{j_{d-L+2}} = \cdots = \lambda_{j_d} = 2$
{Notice that $0 \leq L \leq d-k$}
Set $\gamma_1 = d$, $\gamma_2 = k = 2$
if $L > 0$ then
    for $p = 1$ to $L$ do
        Using the equations corresponding to the $\gamma_2$ rows of type $d-L+p$, decode the
        variables indexed by $\mathcal{I}\left(d-L+p\right)$
        Set $\gamma_1 = \gamma_1 - 1$
    end for
end if
{At this point, $\gamma_1 = d - L$}
if $\lambda_{j_k} > \gamma_1$ then
    Declare failure, exit
else
    Append $T$ with $\gamma_1 - \lambda_{j_k}$ additional rows of type $k = 2$ with non-zero coefficients
    as the rows of $\hat{\Psi}$ that are not present in the $\lambda_{j_2}$ rows of type 2
    Decode the un-decoded variables from $Y[\mathcal{I}\left(2\right)]$
    $\gamma_1 = \gamma_1 - 1$, $\gamma_2 = \gamma_2 - 1$
    {At this point, $\gamma_1 = d - L - 1$ and $\gamma_2 = 2 - 1 = 1$}
    Decode for the remaining symbols from $Y[\mathcal{I}\left(j_{k+1}\right)], Y[\mathcal{I}\left(j_{k+2}\right)], \ldots, Y[\mathcal{I}\left(j_{d-L}\right)]$
    Append $T$ with a row of type 1 with non-zero coefficients as the first row of $\hat{\Psi}$
    Decode the un-decoded variables from $Y[\mathcal{I}\left(1\right)]$
end if
```

Case 2: $k = 2$.

We present the algorithm for successive decoding as algorithm 3.

First, we prove the correctness of the algorithm.

Claim 4. *If algorithm 3 does not report a failure, it finds a solution to $Z = T'Y$, where the construction of T' is described in the algorithm.*

Proof: Notice that there are $k = 2$ elements each in sets $\mathcal{I}\left(l\right)$ for $k+1 \leq l \leq d$. For the L types, j_{d-L+1} through j_d, there are two rows present in T. Thus, all the variables from $\mathcal{I}\left(j_l\right)$ for $d-L+1 \leq l \leq d$ are decoded at the end of first **for** loop.

The number of un-decoded variables in each of group I types is $d - L$. Algorithm will append L rows of type $k = 2$ and decode all the un-decoded variables indexed by $\mathcal{I}\,(k = 2)$ at line 16. Thus, there remains only one un-decoded element each of types j_{k+1} through j_{d-L}, which will be decoded in line 19. At this point, there remains only single un-decoded variable from type 1, ant it is decoded as the final step.

Essentially, algorithm successively decodes all the d types in the following order (j_{d-L+1}, $\ldots$, j_d), ($j_{(k=2)}$), ($j_{k+1}, \ldots, j_{d-L+1}$), j_1, which covers all the B variables. ■

Claim 5. *If $|\mathcal{G}'| \leq d + k - 3$, algorithm 3 always succeeds.*

Proof: The proof is by contradiction. Suppose $|\mathcal{G}'| \leq d + k - 3 = d - 1$ and the algorithm fails. The only possibility of failure is on line 13. The total number of rows in T that have been considered till this point is as follows.

$$\sum_{m=d-L+1}^{d} \lambda_{j_m} + \lambda_{j_2} \overset{(a)}{\geq} 2L + (d - L + 1)$$

$$\therefore \quad \sum_{m=d-L+1}^{d} \lambda'_{j_m} + L + \lambda_{j_2} + 1 \geq d + L + 1, \tag{3.26}$$

$$\therefore \quad \sum_{m=d-L+1}^{d} \lambda'_{j_m} + \lambda_{j_2} \geq d, \tag{3.27}$$

where (a) follows because failure implies $\lambda_{j_2} > d - L$. To get (3.26), we use $\lambda_{j_l} = \lambda'_{j_l} + 1$ for each $l \in [d]$. Now, we can write

$$\begin{aligned} |\mathcal{G}'| &= \sum_{l=1}^{d} \lambda'_{j_l} \\ &\geq \sum_{m=d-L+1}^{d} \lambda'_{j_m} + \lambda_{j_2} \\ &\overset{(b)}{\geq} d, \end{aligned} \tag{3.28}$$

where (b) is due to (3.27). However, $|\mathcal{G}'| \geq d$ is a contradiction and the proof follows. ■

Case 3: $k = 1$.

Notice that for $k = 1$, Eve gets the same degrees of freedom as a data collector (which accesses to k nodes to recover the file). Therefore, it is not possible to achieve any form of security since, similar to the data collector, Eve can also decode the complete file

This completes the proof of Theorem 5. ■

Example: For our running example, consider one possible matrix T as shown in (3.29). To prove that T is full-rank, we show that it is possible to append some rows to T in such a way that successive decoding is possible. First, append two rows of type 3 and decode for the variables indexed by $\mathcal{I}(3) = \{3, 6, 8, 9\}$. Then, using the two rows of type 4, decode for the variables indexed by $\mathcal{I}(4) \setminus ((\mathcal{I}(3) \cap \mathcal{I}(4))) = \{4, 7\}$. Next, using the two rows of type 2, decode for the variables indexed by $\mathcal{I}(2) \setminus ((\mathcal{I}(2) \cap \mathcal{I}(3)) \cup (\mathcal{I}(2) \cap \mathcal{I}(4))) = \{2, 5\}$. Finally, using a row of type 1, decode for the variable indexed by $\mathcal{I}(1) \setminus ((\mathcal{I}(1) \cap \mathcal{I}(3)) \cup (\mathcal{I}(1) \cap \mathcal{I}(4)) \cup (\mathcal{I}(1) \cap \mathcal{I}(2))) = \{1\}$. The non-zero coefficients of the appended rows are specified by the appropriate rows of $\hat{\Psi}$, and the successive decoding is feasible due to the property that any square sub-matrix of the Cauchy matrix $\tilde{\Psi} = \begin{bmatrix} \Psi \\ \hat{\Psi} \end{bmatrix}$ is non-singular.

Remark 13. *In [37], it is shown that, when Eve observes l nodes, the PM-MBR codes using Cauchy matrix as their encoding matrix are weakly secure against $k - l - 1$ guesses without any outer code. Thus, for $l = 1$, it is shown that PM-MBR codes are secure against $k - 2$ guesses. Our proposed encoding enhances the level of security to $d + k - 4$ guesses, which is an improvement of $d - 2$ for all set of parameters (except for $d = k = 2$). Notice that, for any regenerating code, $d \geq k$. Thus, for large k, the enhancement achieved by the proposed scheme is roughly twofold in terms of the number of guesses that Eve can make and still cannot learn anything about any single message symbol.*

$$T = \begin{bmatrix} 0 & \hat{\Psi}(2,1) & 0 & 0 & \hat{\Psi}(2,2) & \hat{\Psi}(2,3) & \hat{\Psi}(2,4) & 0 & 0 \\ 0 & 0 & \hat{\Psi}(3,1) & 0 & 0 & \hat{\Psi}(3,2) & 0 & \hat{\Psi}(3,3) & \hat{\Psi}(3,4) \\ 0 & 0 & 0 & \hat{\Psi}(1,1) & 0 & 0 & \hat{\Psi}(1,2) & 0 & \hat{\Psi}(1,3) \\ \Psi(e,1) & \Psi(e,2) & \Psi(e,3) & \Psi(e,4) & 0 & 0 & 0 & 0 & 0 \\ 0 & \Psi(e,1) & 0 & 0 & \Psi(e,2) & \Psi(e,3) & \Psi(e,4) & 0 & 0 \\ 0 & 0 & \Psi(e,1) & 0 & 0 & \Psi(e,2) & 0 & \Psi(e,3) & \Psi(e,4) \\ 0 & 0 & 0 & \Psi(e,1) & 0 & 0 & \Psi(e,2) & 0 & \Psi(e,3) \end{bmatrix} \tag{3.29}$$

For the MBR case, we carefully designed the outer code taking into account a given family of regenerating codes, namely, product-matrix codes. Essentially, we presented a joint design of the (inner) storage code and the (outer) coset code to achieve weak security. However, jointly designing the codes requires that the user, who preprocess and stores the files, should know the underlying storage code in order to design the (outer) linear transformation for achieving weak security. In many practical scenarios, such as storing the files on third party cloud storage system, it may not be possible for the user to know the underlying storage code. This motivates us to investigate universal outer codes as described in the following section.

3.4 Universally Weakly Secure Coset Codes for MSR Codes

We seek to answer the following question: while designing a weakly secure storage system, is it possible to separate the outer code design from the storage code design? In other words, we would like to develop a *universal* approach to weak security that allows independent design of the storage code and the coset code. We say that an outer code is *universal* if it can weakly secure *any* (inner) regenerating code.

Note that the notion of universal security was proposed in [39, 47] for secure network coding. An outer code is said to be universal if it can be designed completely independently of the network code. Silva and Kschischang [39] proved existence of universal outer codes for weakly securing any network code. Focussing on the paradigm of strong secrecy in [47], they presented explicit universal outer codes based on Maximum Rank-Distance (MRD) codes [44]. Kurihara *et al.* [52] extended the techniques of [47] to construct universal outer codes for weakly securing any network code. The outer code construction in [52] was adapted to the distributed storage setting by Kurihara and Miyake in [38]. However, the main drawback of using a rank-metric code is that the required field size is significantly large. In particular, the universal outer code of [38] requires the field size of q^{2B}, where q is the field size of the underlying storage code and B is the total number of information symbols stored.

Our goal is to construct universal outer coding schemes that require smaller field size. Towards that, we ask the following question. Suppose we know the operating point of an underlying regenerating codes, say MSR or MBR, is it possible to design an outer code that universally achieves weak security for any regenerating code operating at that point such that the field size is lower? We answer this question affirmatively, and present a smaller field size universal outer code that can weakly secure any MSR code.

Our Contributions: First, we present a construction of universal outer code that can achieve weak security of individual symbols in any (n, k) MSR code against an eavesdropper that can observe any $k - 1$ storage nodes. The required field size of the scheme is $O(B^k)$, where B is the number of stored information symbols. Next, we present a construction of universal outer code that can achieve weak security with maximum possible g in any MSR code against an eavesdropper that can observe any single storage node. The required field size of the scheme is $O(B^\alpha)$, where B is the total number of stored information symbols and α is the number of symbols stored on each node.

Remark 14. *As noted in [39], g-weak security is equivalent to perfect security of a message $S' = S_{\mathcal{G}} \subset S$ for any $\mathcal{G}$ of size up to g (see* (3.6)*). In particular, if we treat $S_{\mathcal{G}}$ as a message and the rest of the symbols as random keys (for any $\mathcal{G}$ of size up to g), then* (3.6) *is equivalent to the perfect secrecy of $S_{\mathcal{G}}$. Therefore, it is possible to store B symbols with g-weak security against an eavesdropper observing any μ coded symbols, only if $g + 1 \leq B - \mu$.*

Note that, using (3.6) and (3.7), it follows that a universal coset code H ensures that

$$\text{rank} \begin{bmatrix} H_{\mathcal{G}} \\ G' \end{bmatrix} = \text{rank}\, H_{\mathcal{G}} + \text{ rank}\, G', \tag{3.30}$$

for every $\mathcal{G} \subset [B_s]$ such that $|\mathcal{G}| \leq g$ and for every storage code G. (Recall that G' is a $\mu \times B$ sub-matrix of G corresponding to μ eavesdropped symbols.)

Such a coset code H was first constructed in [39] for $g \leq 2$ using a rank-metric code over $\mathbb{F}_{q^B}$ to secure any G over $\mathbb{F}_q$ with application to network coding. This construction was extended for $g \leq B_s - \mu$ in [52] again using rank-metric codes, requiring the field size of q^{2B}, where q is the field size for the entries of G. The authors of [38] adapted the construction of [52] to weakly secure any regenerating code. The main drawback of such an outer code based on a rank-metric code is its high field size. Instead of securing any regenerating code, we restrict our attention to the class of MSR codes, and present universal outer code constructions over small field size in the next section.

In the following, we present constructions for universal outer codes to achieve g-weak security in any (n, k, d, α, β)-MSR code. For simplicity, we follow (3.6) as the definition of g-weak security. We consider the following two scenarios: (i) maximum ℓ and minium g, *i.e.*, $\ell = k-1$ and $g = 1$, and (ii) minimum ℓ and maximum g, *i.e.*, $\ell = 1$ and $g = B-\alpha$. We assume that the user only knows the code parameters n, k, d, α, β, and q. In addition, we assume that the encoding of the MSR code is systematic. We begin with setting up necessary notation for MSR codes.

Notation for MSR Codes: Consider an (n,k,d,α,β)-MSR code $\mathcal{C}$ over $\mathbb{F}_q$, storing $B = k\alpha$ information symbols (see Sec. 3.2.1). Let $G = \begin{bmatrix} G_1^T & G_2^T & \cdots & G_n^T \end{bmatrix}^T$ be an $n\alpha \times B$ generator matrix of $\mathcal{C}$, where G_i is an $\alpha \times B$ matrix corresponding to the symbols stored on node i. We refer to G_i as a generator matrix of node i. Let us denote G_i as $G_i = [G_{i,1} \ G_{i,2} \ \cdots \ G_{i,\alpha}]$, where $G_{i,j}$ is an $\alpha \times \alpha$ matrix.

We assume that G is in systematic form, and the first k nodes are systematic. In other words, we have $G_{i,i} = I_\alpha$ and $G_{i,j} = 0_\alpha$ for $1 \leq i,j \leq k$ such that $i \neq j$, where I_α is an $\alpha \times \alpha$ identity matrix and 0_α is an $\alpha \times \alpha$ zero matrix.

For any matrix (or vector) W with $B = k\alpha$ columns, we refer to the α columns of W indexed from $(j-1)\alpha + 1$ through $j\alpha$, as the j-th *thick-column* of W $(1 \leq j \leq k)$.

3.4.1 Construction for $\ell = k - 1$ and $g = 1$

Note that $\ell = k - 1$ is the maximum possible strength that Eve can have for an (n,k)-MSR code, as any k nodes recover the entire stored data. The motivation behind $g = 1$ is to protect every individual file, which usually carry meaningful information. The idea for constructing H is to begin with a Vandermonde matrix over some base field and then scale some of its appropriately chosen columns by elements lying in an extension field. The details are given in the following.

Construction 5. *Consider the parameters of an MSR code as n, k, d, α, β and q. Choose q_r as the smallest power of q greater than equal to $B = k\alpha$. For $1 \leq i, j \leq B$, choose the entries $h_{i,j}$ of H as follows:*

$$h_{i,j} = \begin{cases} \omega^{\frac{j}{\alpha}} \beta_j^{i-1} & \textit{if } \alpha \mid j, \\ \beta_j^{i-1} & \textit{otherwise}, \end{cases} \tag{3.31}$$

where $\beta_1, \cdots, \beta_B$ *are* B *distinct elements of* $\mathbb{F}_{q_r}$*, and* ω *is a primitive element of* $\mathbb{F}_{q_r^{k+1}}$.

Next, we show that the above construction can universally achieve 1-weak security for any MSR code for $\ell = k-1$.

Theorem 6. *The outer coset code of Construction 5 can be used universally with any* (n,k)*-MSR code to store* $B_s = k\alpha$ *symbols over* $\mathbb{F}_{q_r^{k+1}}$ *with* 1*-weak security against an eavesdropper observing any* $\ell = k-1$ *nodes, each storing* α *coded symbols.*

Proof. First, note that, since H is a $B \times B$ Vandermonde matrix with some of its columns scaled, it is non-singular, resulting in $B_s = B = k\alpha$.

Next, we prove the 1-weak security. Let H_i denote the i-th row of H. Let the set of nodes accessed by Eve be $\mathcal{L} = \{i_1, \cdots, i_\ell\}$ and let $C_{\mathcal{L}} = \{C_i : i \in \mathcal{L}\}$. For $g = 1$, we want to prove $I\left(S_i; C_{\mathcal{L}}\right) = 0$ for every $i \in [B]$ (see (3.6)). From (3.7), this is equivalent to showing that, for every $i \in [B]$, the following matrix is full-rank:

$$T = \begin{bmatrix} H_i \\ G' \end{bmatrix} = \begin{bmatrix} H_i \\ G_{i_1} \\ G_{i_2} \\ \vdots \\ G_{i_\ell} \end{bmatrix} \tag{3.32}$$

Towards proving this, we consider the following two cases.

Case 1: $\mathcal{L} \subset [k]$. In other words, all the nodes Eve observes are systematic. Let $j = [k] \setminus \mathcal{L}$. Then, the j-th thick-column of G' is zero. Clearly, H_i cannot be in the row space of G'.

Case 2: $\mathcal{L} \not\subset [k]$. In this case, at least one parity node is eavesdropped. Arbitrarily choose an index j such that $j \in [k] \setminus \mathcal{L}$. Note that there are at least two such systematic nodes not in $\mathcal{L}$. Due to the reconstruction property of MSR codes, $\begin{bmatrix} G' \\ G_j \end{bmatrix}$ should be invertible. As the j-th thick-column of G_j is an identity matrix and all its other thick-columns are zero, this implies that the $(k-1)\alpha \times (k-1)\alpha$ matrix, say G'', formed by all the thick-columns of G' except the j-th one is invertible. Thus, we can perform row operations on T to obtain the following matrix:

$$T_1 = \begin{bmatrix} H_i \\ (G'')^{-1}G' \end{bmatrix}.$$

It is easy to see that, after reordering the rows and columns of T_1, we can get

$$T_2 = \begin{bmatrix} H_{i,1} & H_{i,2} & \cdots & H_{i,k-1} & H_{i,k} \\ I_\alpha & 0_\alpha & \cdots & 0_\alpha & P_{1,\alpha} \\ 0_\alpha & I_\alpha & \cdots & 0_\alpha & P_{2,\alpha} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_\alpha & 0_\alpha & \cdots & I_\alpha & P_{k-1,\alpha} \end{bmatrix}, \tag{3.33}$$

where $H_{i,j} = [\beta^{i-1}_{(j-1)\alpha+1}\, \beta^{i-1}_{(j-1)\alpha+2} \cdots \beta^{i-1}_{j\alpha-1}\, \omega^j \beta^{i-1}_{j\alpha}]$, and $P_{i,j} \in \mathbb{F}^{\alpha\times\alpha}_{q_r}$. Now, we obtain a square matrix T_3 by appending T_2 with the $(\alpha-1)\times B$ matrix $T_2' = \begin{bmatrix} 0_{(\alpha-1)\times(k-1)\alpha} & I_{\alpha-1} & 0_{(\alpha-1)\times 1} \end{bmatrix}$, where $0_{t\times m}$ is a $t \times m$ all-zero matrix and $I_{\alpha-1}$ is an $(\alpha-1)\times(\alpha-1)$ identity matrix.

Using the identity matrix $I_{\alpha-1}$ in T_2', we eliminate all but the last entry in the k-th thick-column of T_2 to obtain the following matrix:

$$T_4 = \begin{bmatrix} H_{i,1} & \cdots & H_{i,k-1} & 0_{\alpha\times(\alpha-1)} & \omega^j \beta_{j\alpha}^{i-1} \\ I_\alpha & \cdots & 0_\alpha & 0_{\alpha\times(\alpha-1)} & p_{1,\alpha} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0_\alpha & \cdots & I_\alpha & 0_{\alpha\times(\alpha-1)} & p_{k-1,\alpha} \\ 0_{(\alpha-1)\times\alpha} & \cdots & 0_{(\alpha-1)\times\alpha} & I_{\alpha-1} & 0_{(\alpha-1)\times 1} \end{bmatrix}, \tag{3.34}$$

for some $p_{i,\alpha} \in \mathbb{F}_{q_r}^{\alpha\times 1}$, $1 \leq i \leq k-1$.

Now, the determinant of T_4 can be written as a polynomial in ω as follows:

$$det(T_4) = \omega^j \beta_{j\alpha}^{i-1} + \cdots . \tag{3.35}$$

Note that $\det(T_4)$ is a non-zero polynomial in ω with coefficients in $\mathbb{F}_{q_r}$, *i.e.*, $\det(T_4) \in \mathbb{F}_{q_r}[\omega]$. Further, the degree of this polynomial is at most k. Since ω is a primitive element of $\mathbb{F}_{q_r^{k+1}}$, the degree of its minimal polynomial is $k+1$. Thus, ω cannot be a root of any polynomial in $\mathbb{F}_{q_r}$ of degree at most k. Hence, we have $det(T_4) \neq 0$. Therefore, T_4 is non-singular, and it follows that T must be full-rank. □

Field Size Comparison: It was shown in [39] a parity-check matrix H of a rank-metric code, in particular a Gabidulin code [44], can be used to achieve g-weak security for $g \leq 2$. The field size requirement of such a code is q^B, where q is the field of the underlying storage code and $B = k\alpha$ is the total number of information symbols. Since $q \geq n$ for any known MSR code (see *e.g.*, [53, 54], and references therein), the required field size is $O(n^{k\alpha})$.

The proposed construction operates over the field size of q_r^{k+1}, where q_r is the smallest power of q greater than or equal to B. Assuming that $q = O(n)$ and $B = k\alpha > q$, the proposed construction requires the field size of at most $O\left((nk\alpha)^{k+1}\right)$. Note that for high-rate MSR codes, the best known codes have α to be exponential in k, and it is shown that α needs to be at least exponential in $\sqrt{k}$ [55]. Thus, a rank-metric code based outer code would require significantly larger field size as compared to the proposed scheme.

3.4.2 Construction for $\ell = 1$ and $g = B - \alpha$

Note that $B - \alpha$ is the maximum value of g, as Eve observes α symbols when $\ell = 1$ (see Remark 14). The idea of constructing H is similar to Construction 5. In this case, we begin with a Cauchy matrix over some base field and scale its first α columns by a primitive element of an extension field. The details are as follows.

Construction 6. *Consider the parameters of an MSR code as n, k, d, α, β and q. Choose q_r as the smallest power of q greater than or equal to $2B$, where $B = k\alpha$. Construct H as the product of two matrices as*

$$H = H'W', \tag{3.36}$$

where H' is a $B \times B$ Cauchy matrix with each entry chosen from $\mathbb{F}_{q_r}$, and W' is a $B \times B$ identity matrix with its first α columns scaled by $1/\omega$. Here, ω is a primitive element of the extension field $\mathbb{F}_{q_r^{\alpha+1}}$. We can view H as follows.

$$H = \begin{bmatrix} \frac{h_{1,1}}{\omega} & \cdots & \frac{h_{1,\alpha}}{\omega} & h_{1,\alpha+1} & \cdots & h_{1,B} \\ \frac{h_{2,1}}{\omega} & \cdots & \frac{h_{2,\alpha}}{\omega} & h_{2,\alpha+1} & \cdots & h_{2,B} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{h_{B,1}}{\omega} & \cdots & \frac{h_{B,\alpha}}{\omega} & h_{B,\alpha+1} & \cdots & h_{B,B} \end{bmatrix}, \tag{3.37}$$

where $[h_{i,j}]$, $1 \leq i, j \leq B$, is a Cauchy matrix.

Next, we show that the above construction can universally achieve $(B - \alpha)$-weak security for MSR codes when $\ell = 1$.

Theorem 7. *The outer coset code of Construction 6 can be used universally with any (n, k) MSR code to store $B_s = k\alpha$ symbols over $\mathbb{F}_{q_r^{\alpha+1}}$ with g-weak security for $g = B - \alpha$ against an eavesdropper observing any $\ell = 1$ node storing α coded symbols.*

Proof. First, it is easy to see that H is non-singular, as it is a $B \times B$ Cauchy matrix with some of its columns scaled by ω. Thus, we have $B_s = B$.

Next, we want to show that matrix $T = \begin{bmatrix} H_{\mathcal{G}} \\ G_e \end{bmatrix}$ is full-rank, where G_e is the generator matrix of the observed node and $H_{\mathcal{G}}$ consists of $B - \alpha$ rows of H (see Lemma 2). Consider the case when Eve observes one of the systematic nodes. Since H' is a Cauchy matrix, any of its square sub-matrices is full-rank. Using this property, it is easy to show that T will be full-rank.

Suppose Eve observes a parity node j, $k + 1 \leq j \leq n$. Note that, for any parity node $k + 1 \leq j \leq n$, one can easily show that the $\alpha \times \alpha$ block $G_{j,1}$ is full-rank as follows. Suppose that a data collector downloads from parity node j, $k+1 \leq j \leq n$, and systematic nodes 2 through k. Since any k out of n nodes allow reconstructing the set of B files, $G_{j,1}$ must be full-rank.

Since $G_{j,1}$ is full-rank, we pre-multiply G_j in T by $G_{j,1}^{-1}$. Then, by multiplying each of the first α columns by ω, we can transform T to the matrix T' shown in (3.38), for some $p_{i,j} \in \mathbb{F}_{q_r}, 1 \leq i \leq \alpha, 1 \leq j \leq B - \alpha$.

$$T' = \begin{bmatrix} H_{\mathcal{G}} \\ G_{j,1}^{-1} G_j \end{bmatrix} (W')^{-1} = \begin{bmatrix} h_{j_1,1} & h_{j_1,2} & \cdots & h_{j_1,\alpha} & h_{j_1,\alpha+1} & h_{j_1,\alpha+2} & \cdots & h_{j_1,B} \\ \vdots & & \ddots & & \vdots & & \ddots & \vdots \\ h_{j_g,1} & h_{j_g,2} & \cdots & h_{j_g,\alpha} & h_{j_g,\alpha+1} & h_{j_g,\alpha+2} & \cdots & h_{j_g,B} \\ \omega & 0 & \cdots & 0 & p_{1,1} & p_{1,2} & \cdots & p_{1,B-\alpha} \\ 0 & \omega & \cdots & 0 & p_{2,1} & p_{2,2} & \cdots & p_{2,B-\alpha} \\ \vdots & & \ddots & & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \omega & p_{\alpha,1} & p_{\alpha,2} & \cdots & p_{\alpha,B-\alpha} \end{bmatrix} \tag{3.38}$$

Now, the determinant of T' can be written as a polynomial in ω as follows:

$$det(T') = \left[det\left(H'_{\mathcal{G}}(\alpha + 1 : B)\right)\right] \omega^{\alpha} + \cdots, \tag{3.39}$$

where $H'_{\mathcal{G}}(\alpha + 1 : B)$ is the $(B - \alpha) \times (B - \alpha)$ sub-matrix of $H'_{\mathcal{G}}$ formed by its last $B - \alpha$ columns. Since H' is a Cauchy matrix, $det\left(H'_{\mathcal{G}}(\alpha + 1 : B)\right) \neq 0$. Hence, $det(T')$ is a non-zero polynomial in ω with coefficients in $\mathbb{F}_{q_r}$, *i.e.*, $det(T'') \in \mathbb{F}_q[\omega]$. Further, $deg(det(T'')) = \alpha$. Since, ω is a primitive element of $\mathbb{F}_{q_r^{\alpha+1}}$, it cannot be a root of a degree α polynomial in $\mathbb{F}_{q_r}[\omega]$. Therefore, $det(T') \neq 0$, and it follows that T is full-rank. □

Field Size Comparison: The universal outer code in [38] based on rank-metric codes achieves g-weak security for any ℓ and maximum possible g. The field size requirement is q^{2B}, where q is the field of the underlying storage code and $B = k\alpha$ is the total number of information symbols. Since $q \geq n$ for any known MSR code (see *e.g.*, [53, 54], and references therein), the required field size is $O(n^{2k\alpha})$.

The field size required for the proposed construction is $q_r^{\alpha+1}$, where q_r is the smallest power of q greater than or equal to $2B$. Assuming that $q = O(n)$ and $2B = 2k\alpha > q$, the proposed construction requires the field size of at most $O\left((nk\alpha)^{\alpha+1}\right)$. Note that for high-rate MSR codes, the best known codes have α to be exponential in k, and it is shown that α needs to be at least exponential in $\sqrt{k}$ [55]. When α is exponential in k, one can verify that the proposed scheme requires a smaller field size than the rank-metric code based scheme of [38] for a wide range of parameters.

3.5 Conclusion

In the first half, we constructed outer codes to achieve weak security in product-matrix MBR codes. Our proposed outer code requires a field size linear in number of nodes (n), and at the same time, enhances the weak security level g by roughly twofold.

In the second half, we focused on the weak security paradigm in which Eve gains no information about any group of g symbols. We proposed a universal outer code that can weakly secure any MSR code. In particular, we considered two scenarios: (i) the eavesdropper has the maximum strength of $\ell = k - 1$, and the weak security level is the minimum $g = 1$; and (ii) the eavesdropper has the minimum strength of $\ell = 1$, but the weak security level is the maximum possible $g = B - \alpha$. Our key idea is to utilize the structure present in the (systematic) generator matrix of an MSR code to construct the outer code. This enabled us to reduce the required field size compared to the standard approaches based on rank-metric codes.

4. CODES WITH UNEQUAL LOCALITY*

4.1 Introduction

Coding for distributed storage has recently attracted significant research attention with a focus on the problem of recovery from storage node failures. The thrust has been on characterizing fundamental limits and designing associated coding schemes for one or more of the following metrics that are crucial in the node repair process: (a) *repair bandwidth* – the amount of data downloaded during failed node repair [8, 57]; (b) disk I/O – the number of bits read from the nodes participating in the repair process [14, 58]; and (c) *repair locality* – the number of nodes participating in the repair process [9, 20].

In this chapter, we focus on the metric of repair locality and a class of codes designed in the context of this metric, known as *locally repairable codes* (LRCs). Consider a block code of length n that encodes k information symbols. A symbol i is said to have *locality* r_i if it can be recovered by accessing r_i other symbols in the code. We say that a code has *information locality* r if each of its k information symbols has locality at most r. Similarly, we say that a code has *all-symbol locality* r if each of its n symbols has locality at most r.

Codes with small locality were introduced in [18, 19] (see also [20]). The study of the locality property was galvanized with the pioneering work of Gopalan *et al.* [9]. One of their key contributions was to establish a trade-off between the minimum distance of a code and its information locality analogous to the classical Singleton bound. In particular, the authors showed that for a (scalar) linear (n, k) code having information locality r, the

*Parts of this chapter are reprinted with permission from [56] "Codes with unequal locality," by S. Kadhe and A. Sprintson, 2016. *In Proceedings of 2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 435-439, July 2016. Copyright © by IEEE.

minimum distance d of the code is upper bounded as

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{4.1}$$

They also demonstrated that the Pyramid code construction in [18] achieves this bound. Since then, a series of results have extended the code distance bound for a given locality for various types of codes along with corresponding *optimal* code constructions achieving the distance bound. We give a brief (far from complete) overview of some of these results below.

Related work: The distance bound was generalized for codes with multiple local parities in [59], universal (scalar/vector linear, nonlinear) codes in [21], universal codes with multiple parities in [22, 60]. An integer programming based bound was established in [23]. Almost all of these works also presented optimal code constructions. Furthermore, a large number of other optimal code constructions have been presented, see *e.g.*, [24, 61, 62, 25, 55, 63, 64, 26, 65, 66]. The notion of locality was extended to multiple recovery groups (also known as, availability) in [67, 68], and for the case of multiple failures, to sequential repair in [69] and hierarchical repair in [70]. The Singleton-like bound was extended to accommodate the alphabet size in [71].

Our contributions: In previous works, the locality of a code is characterized by a single parameter r. Inspired by the notion of *unequal error protection*, we are interested in investigating linear codes, in which, different subsets of symbols possess different values of locality. We refer to such codes as *codes with unequal locality*. For example, consider a $(18, 11)$ code whose 4 information symbols have locality 2, 3 information symbols have locality 3, and 4 information symbols have locality 4 (with no constraint on the locality of parity symbols). In previous works, such a code would be characterized as a code with information locality 4, for which (4.1) gives $d \leq 6$. However, the distance bound given in (4.1) is not always tight for the case of unequal locality. For instance, for the previous code, we will show that $d \leq 5$. Our main goal is to establish a tight upper bound on the minimum distance for codes with unequal locality.

Codes with unequal locality are practically appealing in scenarios when important information symbols, *e.g.*, symbols of *hot data*, need to be repaired quickly; whereas, recovering less important symbols can involve more overhead. Moreover, these types of codes can be useful in reducing download latency for hot data. For instance, references [72, 73] study storage codes from queueing theoretic perspective to analyze download latency.

Our key contributions are summarized as follows. To characterize a code with unequal information locality, we define a notion of *information locality profile* of a code. We say that a code has an information locality profile $\mathbf{k} = \{k_1, \dots, k_r\}$ if it contains k_j information symbols of locality j for $1 \leq j \leq r$. For example, a code having 5 information symbols of locality 2, and 6 information symbols of locality 4 would have an information locality profile $\{0, 5, 0, 6\}$. For scalar linear codes, we establish an upper bound on the minimum distance as a function of information locality profile $\mathbf{k} = \{k_1, \dots, k_r\}$ as follows (Theorem 8).

$$d \leq n - k - \sum_{j=1}^{r} \left\lceil \frac{k_j}{j} \right\rceil + 2. \tag{4.2}$$

We demonstrate that the Pyramid code construction in [18] can be adapted to design unequal locality codes that are distance-wise optimal according to the bound above. This result generalizes the classical result of Gopalan *et al.* [9] for codes with unequal locality.

When parity symbols also have locality constraints, we analogously define the *all-symbol locality profile* of a code. We say that a code has all-symbol locality profile $\mathbf{n} = (n_1, \dots, n_{r_a})$ if it contains n_j symbols of locality j for $1 \leq j \leq r_a$. For instance, consider a $(15, 11)$ code that has 6 symbols of locality 2, 4 symbols of locality 3, and 5 symbols of locality 4. Its all-symbol locality profile would be $(0, 6, 4, 5)$. We establish a tight upper bound on the minimum distance for scalar linear codes with unequal all-symbol locality as (see Theorem 9)

$$d \leq n - k - \sum_{j=1}^{r-1} \left\lceil \frac{n_j}{j+1} \right\rceil - \left\lceil \frac{k - \sum_{j=1}^{r-1} \left(n_j - \left\lceil \frac{n_j}{j+1} \right\rceil \right)}{r} \right\rceil + 2,$$

where $r := 1 + \max \left\{ 1 \leq i \leq r_a : \sum_{j=1}^{i} \left(n_j - \left\lceil \frac{n_j}{j+1} \right\rceil \right) < k \right\}$. We adapt the construction in [24], which uses a maximum rank-distance (MRD) code as an outer code and a maximum distance separable (MDS) code as an inner code, to construct codes with unequal all-symbol locality that are distance-wise optimal with respect to the above bound.

We note that in a parallel and independent work, Zeh and Yaakobi [74] also consider the problem of computing a bound on minimum distance of codes with unequal all-symbol locality, referred in their work as codes with multiple localities. Their bound [74, Theorem 8] holds for a slightly restrictive set of parameters as it assumes $\sum_{j=1}^{r_a-1} j \left\lceil \frac{n_j}{j+1} \right\rceil < k - 1$.

Finally, we introduce a concept of *information locality requirement*. To motivate this, consider a scenario where we need to design a linear code of dimension $k = 11$ such that $\tilde{k}_3 = 5$ information symbols must have locality at most 3, and the remaining $\tilde{k}_4 = 6$ information symbols must have locality at most 4. Collectively, we can specify this as a locality requirement of $\tilde{\mathbf{k}} = \{0, 0, 5, 6\}$. Notice that this is equivalent to a requirement as a code must contain at least 5 symbols of locality up to 3, and at least 11 symbols of locality up to 4. In general, a locality requirement of $\tilde{\mathbf{k}} = \{\tilde{k}_1, \ldots, \tilde{k}_r\}$ means that a code should contain at least $\sum_{i=1}^{i} \tilde{k}_j$ symbols of locality up to i for each $1 \leq i \leq r$, or, in other words, $\tilde{k}_j$ information symbols should have locality at most j.

One can design codes with various information locality profiles that would satisfy this requirement. For example, the locality requirement of $\tilde{\mathbf{k}} = (0, 0, 5, 6)$ is satisfied by locality profiles $(k_1, k_2) = (5, 6)$, $(k_1, k_2, k_3) = (0, 2, 9)$, $(k_1, \ldots, k_4) = (0, 0, 5, 6)$,

$(k_1, \dots, k_4) = (1, 0, 6, 4)$, etc. We ask the following question: Can we find an information locality profile which achieves the maximum value of the minimum distance among all codes which satisfy this locality requirement? We present a simple greedy algorithm to compute such an information locality profile that maximizes the minimum distance for a given information locality requirement. This allows us to construct codes that have maximum minimum distance among all codes that satisfy the locality requirement.

4.2 Preliminaries

Notation: For an integer l, let $[l] = \{1, 2, \dots, l\}$. For a vector $\mathbf{x}$, let $\mathbf{x}(i)$ be its i-th coordinate, and $\mathsf{wt}\left(\mathbf{x}\right)$ be its Hamming weight. For a matrix H, let $\mathsf{Rank}\left(H\right)$ denote the rank of H.

Codes with Locality

Let $\mathcal{C}$ denote a linear $[n, k, d]_q$ code over $\mathbb{F}_q$ with block-length n, dimension k, and minimum distance d. We assume that $\mathcal{C}$ has minimum distance at least two, *i.e.*, $d \geq 2$. The code can be represented by n (column) vectors $C = [\mathbf{c}_1, \dots, \mathbf{c}_n] \in \mathbb{F}_q^k$. Note that the dimension of the row space of C is k. The i-th vector $\mathbf{c}_i$ is referred to as the i-th coordinate of C. For any codeword $\mathbf{c} \in \mathcal{C}$, $\mathbf{c}(i)$ is said to be the i-th symbol of the codeword $\mathbf{c}$. Throughout this chapter, we use the terms symbol and coordinate interchangeably.

We say that the i-th coordinate of a code $\mathcal{C}$ has locality r_i if its value can be recovered from some other r_i coordinates of $\mathcal{C}$. The formal definition of locality is as follows (see [9]).

Definition 6. *[Locality and Repair Group] For $\mathbf{c}_i \in C$, we define the locality of $\mathbf{c}_i$ as the smallest integer r_i such that there exists a set $R\left(i\right) \subset [n] \setminus \{i\}$, $|R\left(i\right)| = r_i$, such that $\mathbf{c}_i = \sum_{l \in R(i)} \lambda_l \mathbf{c}_l$, where $\lambda_l \in \mathbb{F}_q, \lambda_l \neq 0, \forall\, l \in R\left(i\right)$. We say that $R\left(i\right)$ is a repair group of the coordinate $\mathbf{c}_i$, and define $\Gamma\left(i\right) = \{\mathbf{c}_i \cup R\left(i\right)\}$.*

Note that, if the minimum distance of the code is at least two, then every coordinate has locality at most k.

We say that an $[n, k, d]_q$ systematic code has *information locality* r if each of its k information symbols has locality at most r. Similarly, we say that an $[n, k, d]_q$ code has *all-symbol locality* r_a if each of its n symbols has locality at most r_a.

4.3 Codes with Unequal Information Locality

In this section, we are interested in systematic codes, whose information symbols have different locality values. More specifically, information symbols can be partitioned into disjoint subsets in such a way that the symbols in one subset have different locality than the symbols in the other subsets. We can characterize the locality of such codes by listing the locality values for all information symbols. Alternatively, we can consider the list of cardinalities of subsets of all locality values. We call such a list as the information locality profile of the code. Formally, the definition is given below.

Definition 7. *[Information Locality Profile] Given a systematic $[n, k, d]_q$ code $\mathcal{C}$, the information locality profile of $\mathcal{C}$ is defined as a length-r vector $\mathbf{k}(\mathcal{C}) = (k_1, \ldots, k_r)$, where k_j is the number of information coordinates of locality j.*

Note that $r \leq k$, $0 \leq k_j \leq k \; \forall j \in [r]$, $k_r \geq 1$ and $\sum_{j=1}^{r} k_j = k$.

Remark 15. *For a code $\mathcal{C}$ with representation C, we can choose any set of k linearly independent coordinates of C to represent information symbols. Thus, in principle, information locality profile of a code depends upon the particular choice of coordinates for representing information symbols. We always choose the coordinates having smallest locality as information coordinates. More specifically, for $1 \leq j \leq r$, let $C_j \subset C$ be the subset of coordinates having locality j. Set $C_0 = \emptyset$. Let*

$$k_j = \textit{Rank}\left(\cup_{i=0}^{j} C_i\right) - \textit{Rank}\left(\cup_{i=0}^{j-1} C_i\right). \tag{4.3}$$

In other words, $\sum_{i=1}^{i} k_j$ is the rank of the sub-matrix of C formed by the coordinates having locality up to i. Starting with $j = 1$, we choose a subset $I_j \subset C_j$ of k_j linearly independent coordinates to represent k_j information symbols, and continue incrementing j until the total rank is k.

Remark 16. *We can alternatively specify the information locality profile of a systematic code as a length-k vector $\mathbf{r} = \{r_1, \ldots, r_k\}$, where r_j denotes the locality of the j-th information symbol. In the classical notion of locality defined by Gopalan et al. [9], technically, every symbol can have different locality. However, the (information) locality of a code is parameterized by a single value r, which is the largest locality of an (information) symbol. On the other hand, we parameterize the information locality using a length-k vector that specifies the locality of each individual information symbol. We are interested in characterizing a trade-off between the minimum distance of a code and its locality profile vector.*

4.3.1 Bound on the Minimum Distance

Consider a class of systematic linear codes having an information locality profile $\mathbf{k} = \{k_1, \ldots, k_r\}$. In the following, we find an upper bound on the minimum distance as a function of the code length, dimension, and information locality profile.

Theorem 8. *For any linear code with block-length n, dimension k, and information locality profile $\mathbf{k} = (k_1, \ldots, k_r)$, the minimum distance is upper bounded as*

$$d \leq n - k - \sum_{j=1}^{r} \left\lceil \frac{k_j}{j} \right\rceil + 2. \tag{4.4}$$

Proof. We build on the proof technique proposed in [9]. The idea is to construct a large set $S \subseteq C$ such that $\mathsf{Rank}\,(S) \leq k - 1$, and then use the following well known result.

named 4 Construct set $S \subseteq C$ such that $\mathsf{Rank}\,(S) \leq k-1$

```
 1: Let $S_0 = \emptyset$, $i = 1$
 2: **while** $\mathsf{Rank}\,(S_{i-1}) \leq k-2$ **do**
 3:     Pick a coordinate $\mathbf{c}_i \in C \setminus S_i$ having smallest locality
 4:     **if** $\mathsf{Rank}\left(S_{i-1} \cup \mathbf{c}_{\Gamma(i)}\right) < k$ **then**
 5:         Set $S_i = S_{i-1} \cup \mathbf{c}_{\Gamma(i)}$
 6:     **else**
 7:         Pick $\Gamma(i)' \subset \Gamma(i)$ such that $\mathsf{Rank}\left(S_{i-1} \cup \mathbf{c}_{\Gamma(i)'}\right) = k-1$
 8:         Set $S_i = S_{i-1} \cup \mathbf{c}_{\Gamma(i)'}$
 9:     **end if**
10:     Increment $i$
11: **end while**
```

Proposition 1. *([9]) The code $\mathcal{C}$ has minimum distance d if and only if for every $S \subseteq C$ such that $\mathsf{Rank}\,(S) \leq k-1$, we have*

$$|S| \leq n - d. \tag{4.5}$$

Recall that $R(i)$ denotes a repair group of $\mathbf{c}_i$, and we have $|R(i)| = \mathsf{Loc}\,(\mathbf{c}_i)$. Define $\Gamma(i) := \{i \cup R(i)\}$. Further, for any subset $T \subseteq [n]$, define $\mathbf{c}_T = \{\mathbf{c}_i \in C : i \in T\}$.

We use Algorithm 4 to construct a set S such that $\mathsf{Rank}\,(S) < k$. First, note that in line 3, as $\mathsf{Rank}\,(S_{i-1}) \leq k-2$, and there are k (linearly independent) information symbols, there exists a coordinate $\mathbf{c}_i \notin S_{i-1}$.

Our goal is to find a lower bound on $|S|$. Let l be the total number of iterations of Algorithm 4. Observe that $|S| = |S_l|$. Further, the final set S_l has $\mathsf{Rank}\,(S_l) = k-1$. We define the increment in the size and rank of set S_i in the i-th iteration as follows.

$$s_i = |S_i| - |S_{i-1}|, \quad t_i = \mathsf{Rank}\,(S_i) - \mathsf{Rank}\,(S_{i-1})\,. \tag{4.6}$$

Note that

$$|S_l| = \sum_{i=1}^{l} s_i, \quad \mathsf{Rank}\left(S_l\right) = \sum_{i=1}^{l} t_i = k-1. \tag{4.7}$$

We consider two cases depending on whether Algorithm 4 reaches the condition in line 4, *i.e.*, $\mathsf{Rank}\left(S_{i-1} \cup \mathbf{c}_{\Gamma(i)}\right) = k$. We note that the condition can be reached only in the last iteration.

Case 1: Suppose we have $\mathsf{Rank}\left(S_{i-1} \cup \mathbf{c}_{\Gamma(i)}\right) \leq k-1$ throughout. Now, in the i-th iteration, we add $\mathbf{c}_{\Gamma(i)}$ to S. Thus, $s_i \leq \mathsf{Loc}\left(\mathbf{c}_i\right) + 1$. Further, vectors in $\mathbf{c}_{\Gamma(i)} \setminus S_{i-1}$ are such that they yield a (possibly zero) vector in $\langle S_{i-1} \rangle$. Therefore,

$$t_i \leq s_i - 1 \leq \mathsf{Loc}\left(\mathbf{c}_i\right). \tag{4.8}$$

Using this, we can write

$$|S| = \sum_{i=1}^{l} s_i \geq \sum_{i=1}^{l} (t_i + 1) = k - 1 + l, \tag{4.9}$$

where the last equality follows from (4.7).

Lower bounding the number of iterations. Now, to find a lower bound on $|S|$, we find a lower bound on l. Let m be the locality of the last symbol collected by Algorithm 4, where $m \in [r]$. For $1 \leq j \leq m$, let l_j be the number of iterations in which Algorithm 4 picks coordinates of locality j. Note that, if $\mathcal{C}$ does not contain any symbol of a particular locality j, we set $l_j = 0$. Thus, for each j, $0 \leq l_j \leq l$, and $l = \sum_{j=1}^{m} l_j$.

Recall that $C_j \subset C$ is the set of coordinates of locality j (see Remark 15). Since the algorithm collects all coordinates of locality up to j before collecting any coordinate of locality $j+1$ for $1 \leq j \leq m-1$, we have $S_{\sum_{p=1}^{j} l_p} = \cup_{p=1}^{j} C_p$. Therefore, from (4.3), $\mathsf{Rank}\left(S_{l_1}\right) = k_1$ and for $2 \leq j \leq m-1$, $\mathsf{Rank}\left(S_{\sum_{p=1}^{j} l_p}\right) - \mathsf{Rank}\left(S_{\sum_{p=1}^{j-1} l_p}\right) = k_j$. This results in

$$\mathsf{Rank}\left(S_{\sum_{p=1}^{j} l_p}\right) = \sum_{p=1}^{j} k_p, \quad \text{for } 1 \leq j \leq m-1. \tag{4.10}$$

The above two results can be interpreted as follows. The increment in the rank of S by collecting all the coordinates of locality j is k_j for $1 \leq j \leq m-1$. The rank of S, when it contains all the coordinates of locality up to j, is $\sum_{p=1}^{j} k_p$.

When the algorithm terminates, it may not have collected all the coordinates of locality m. Let k'_m be the increment in the rank of S by the coordinates of locality m that are collected by the algorithm. Note that $1 \leq k'_m \leq k_m$.

Note that $\mathsf{Rank}\left(S_l\right) = \mathsf{Rank}\left(S_{\sum_{j=1}^{m-1} l_j}\right) + k'_m$. Using the fact that $\mathsf{Rank}\left(S_l\right) = k-1$ and (4.10), we get $k-1 = \sum_{j=1}^{m-1} k_j + k'_m$. On the other hand, by definition of locality profile vector, we have $\sum_{j=1}^{r} k_j = k$. We consider two cases.

Case (1a): $k_r \geq 21$. Then, it must be that $m = r$ and $k'_m = k_r - 1$ since $1 \leq k'_m \leq k_m$.

Case (1b): $k_r = 1$. Then, it follows that $m = r-1$, and $k'_m = k_{r-1}$ since $1 \leq k'_m \leq k_m$.

In summary, for $1 \leq j \leq r-1$, the increment in the rank of S by collecting the coordinates of locality j is k_j. The increment in the rank of S by locality r coordinates is $k_r - 1$. (Note that this holds for Case (b) as well.) Moreover, for each $1 \leq j \leq r$, when the algorithm is collecting the coordinates of locality j, the rank can increase by at most j in each step (see (4.8)). Therefore, $l_j \geq \left\lceil \frac{k_r-1}{r} \right\rceil$ for $1 \leq j \leq r-1$ and $l_r \geq \left\lceil \frac{k_r-1}{r} \right\rceil$.

Combining this with $l = \sum_{j=1}^{r} l_j$ gives,

$$l \geq \sum_{j=1}^{r-1} \left\lceil \frac{k_j}{j} \right\rceil + \left\lceil \frac{k_r - 1}{r} \right\rceil. \tag{4.11}$$

Substituting this into (4.9), we get

$$|S| \geq k - 1 + \sum_{j=1}^{r-1} \left\lceil \frac{k_j}{j} \right\rceil + \left\lceil \frac{k_r - 1}{r} \right\rceil \tag{4.12}$$

$$\geq k - 2 + \sum_{j=1}^{r} \left\lceil \frac{k_j}{j} \right\rceil . \tag{4.13}$$

Case 2: In the last step, we get $\textsf{Rank}\left(S_{l-1} \cup \mathbf{c}_{\Gamma(l)}\right) = k$. For $1 \leq i \leq l-1$, in the i-th iteration, we add $\mathbf{c}_{\Gamma(i)}$. Thus, $s_i \leq \textsf{Loc}\left(\mathbf{c}_i\right) + 1$. Further, vectors in $\mathbf{c}_{\Gamma(i)} \setminus S_{i-1}$ are such that they yield a (possibly zero) vector in $\langle S_{i-1} \rangle$. Therefore, for $1 \leq i \leq l-1$, we get $t_i \leq s_i - 1 \leq \textsf{Loc}\left(\mathbf{c}_i\right)$. In the last step l, we add $\mathbf{c}_{\Gamma(l)'} \subset \mathbf{c}_{\Gamma(l)}$. This increments $\textsf{Rank}\left(S\right)$ by $t_l \geq 1$ (since $\textsf{Rank}\left(S_{l-1}\right) \leq k-2$), and $|S|$ by $s_l \geq t_l$. Therefore, we have

$$|S| = \sum_{i=1}^{l} s_i \geq \sum_{i=1}^{l-1} (t_i + 1) + t_l = k - 1 + l - 1, \tag{4.14}$$

the last equality follows from (4.7).

Lower bounding the number of iterations. Similar to Case 1, in each iteration i (including the last one), we have $t_i \leq \textsf{Loc}\left(\mathbf{c}_i\right)$. The only difference from Case 1 is that S accumulates total rank of k instead of $k-1$. Therefore, to lower bound l, we can use the same arguments as in Case 1 along with $\textsf{Rank}\left(S_l\right) = k$ to obtain $l \geq \sum_{j=1}^{r-1} \left\lceil \frac{k_j}{j} \right\rceil + \left\lceil \frac{k_r}{r} \right\rceil$ in place of (4.11). Substituting this into (4.14) yields $|S| \geq k - 2 + \sum_{j=1}^{r} \left\lceil \frac{k_j}{j} \right\rceil$ (which is same as (4.13)).

Finally, noting that $|S| \leq n - d$ from Fact 1 and using this lower bound on $|S|$ gives (4.4). □

4.3.2 Code Construction: Pyramid Codes

We show that the *parity splitting* construction of the Pyramid codes due to [18] can be adapted to obtain unequal information locality codes that are optimal with respect to (4.4).

Consider an information locality profile $\mathbf{k}$. Let $\{j_1, \ldots, j_m\}$ with $j_1 < \cdots < j_m$ be the m $(\leq r)$ values of locality such that $k_{j_p} > 0$ for each $p \in [m]$. We begin with a $(k+d-1, k, d)$ systematic maximum distance separable (MDS) code $\mathcal{C}'$, such as Reed-Solomon codes. Let the coordinates of $\mathcal{C}'$ be $C' = [\mathbf{e}_1, \ldots, \mathbf{e}_k, \mathbf{p}_0, \ldots, \mathbf{p}_{d-2}]$, where $\mathbf{e}_j$ is the j-th column of a $k \times k$ identity matrix, and $\mathbf{p}_j$ for $0 \leq j \leq d-2$ are the columns representing the parity symbols.

We partition the set $[k]$ into m disjoint subsets $S_1, \ldots, S_m$ such that $|S_p| = k_{j_p}$ for each $p \in [m]$. Next, we partition each subset S_p into $l_p = \left\lceil \frac{k_{j_p}}{j_p} \right\rceil$ disjoint subsets, $S_{p,i}$, each of size at most j_p. That is, $S_p = \cup_{i=1}^{l_p} S_{p,i}$. For a vector $\mathbf{x}$ of length k, and a set $S \subseteq [k]$, let $\mathbf{x}|_S$ denote the $|S|$-dimensional restriction of $\mathbf{x}$ to the coordinates in set S. Then, we define the systematic code $\mathcal{C}$ with the following representation.

$$
\begin{aligned}
C = \Big[& \mathbf{e}_1, \ldots, \mathbf{e}_k, \mathbf{p}_0|_{S_{1,1}}, \ldots, \mathbf{p}_0|_{S_{1,l_1}}, \\
& \mathbf{p}_0|_{S_{2,1}}, \ldots, \mathbf{p}_0|_{S_{2,l_2}}, \ldots, \mathbf{p}_0|_{S_{m,1}}, \ldots, \mathbf{p}_0|_{S_{m,l_m}}, \\
& \mathbf{p}_1, \ldots, \mathbf{p}_{d-2} \Big]. \qquad (4.15)
\end{aligned}
$$

Note that we have *split* the parity $\mathbf{p}_0$ into $\sum_{p=1}^{m} \left\lceil \frac{k_{j_p}}{j_p} \right\rceil = \sum_{j=1}^{r} \left\lceil \frac{k_j}{j} \right\rceil$ parities. Therefore, $n = k + d - 2 + \sum_{j=1}^{r} \left\lceil \frac{k_j}{j} \right\rceil$. It is easy to verify that parity splitting does not affect the distance, and hence, the code $\mathcal{C}$ has distance d. Since $\mathcal{C}'$ is an MDS code, we have $\mathsf{wt}(\mathbf{p}_0) = k$. Therefore, for each $p \in [m]$, the set of k_{j_p} information coordinates and $\left\lceil \frac{k_{j_p}}{j_p} \right\rceil$ parity coordinates have locality at most j_p. Similar to the classical Pyramid codes in [18], the last $d-2$ parity symbols may have locality as large as k.

4.4 Codes with Unequal All Symbol Locality

In this section, we extend the notion of information locality profile to accommodate the codes whose parity symbols also have locality constraints. In this case, all code symbols (information as well as parity) can be partitioned into disjoint subsets according to their locality, with maximum locality $r_a < k$. (In fact, a code need not be systematic.) We define all-symbol locality profile of a code as follows.

Definition 8. *[All-Symbol Locality Profile] Given an $[n, k, d]_q$ code $\mathcal{C}$, the all-symbol locality profile of $\mathcal{C}$ is defined as a length-r_a vector $\mathbf{n}(\mathcal{C}) = (n_1, \ldots, n_{r_a})$, where n_j is the number of coordinates of locality j for $1 \leq j \leq r_a$.*

Note that $r_a \leq k$, $0 \leq n_j \leq n \ \forall j \in [r_a]$, $n_{r_a} \geq 1$ and $\sum_{j=1}^{r_a} n_j = n$.

Remark 17. *We can alternatively specify the all symbol locality profile of a $\mathcal{C}$ as a length-n vector $\mathbf{r}(\mathcal{C}) = \{r_1, \ldots, r_n\}$, where r_i is the locality of the i-th coordinate of $\mathcal{C}$. Note that $1 \leq r_i \leq k$ for each $i \in [n]$, assuming $d \geq 2$.*

Remark 18. *For a code $\mathcal{C}$ with representation C, let $C_j \subset C$ be the subset of coordinates having locality j for $j \in [r_a]$. If $n_j = 0$ for some j, then we set $C_j = \emptyset$. For $1 \leq j \leq r_a$, we define*

$$k_j := \textsf{Rank}\left(\cup_{i=0}^{j} C_i\right) - \textsf{Rank}\left(\cup_{i=0}^{j-1} C_i\right), \tag{4.16}$$

where we set $C_0 = \emptyset$. Define $r = \max\{j : k_j > 0\}$. Then, $\mathbf{k} = (k_1, \ldots, k_r\}$ gives the information locality profile of $\mathcal{C}$. Note that codes with different information locality profiles can have the same all-symbol locality profile.

4.4.1 Bound on the Minimum Distance

Note that codes with unequal all-symbol locality are a special class of codes with unequal information locality. Therefore, the minimum distance upper bound in (4.4) holds for an all-symbol locality code having information locality profile $\mathbf{k}$. As noted in Remark 18, codes can have different information locality profiles for a given all-symbol locality profile. The upper bound in (4.4) obtained using information locality profile may not be tight for certain information localities. Therefore, we establish an upper bound on the minimum distance as a function of the all-symbol locality profile.

Theorem 9. *Consider a code $\mathcal{C}$ with all-symbol locality profile* $\mathbf{n} = (n_1, \ldots, n_{r_a})$*. Let* $r' = \max\left\{1 \leq i \leq r_a : \sum_{j=1}^{i}\left(n_j - \left\lceil\frac{n_j}{j+1}\right\rceil\right) < k\right\}$*. Let* $r = \min\{r'+1 \leq j \leq r_a : n_j \geq 2\}$*. Then, we have*

$$d \leq n - k - \sum_{j=1}^{r-1}\left\lceil\frac{n_j}{j+1}\right\rceil - \left\lceil\frac{k - \sum_{i=1}^{r-1}\left(n_j - \left\lceil\frac{n_j}{j+1}\right\rceil\right)}{r}\right\rceil + 2 \tag{4.17}$$

Proof. Similar to information locality case, we consider Algorithm 4 to find a set $S \subset C$ such that $\mathsf{Rank}\,(S) \leq k-1$.

Recall that $C_j \subset C$ is a subset of coordinates of locality j. Let $k_j = \mathsf{Rank}\left(\cup_{i=0}^{j}C_i\right) - \mathsf{Rank}\left(\cup_{i=0}^{j-1}C_i\right)$, where we define $C_0 = \emptyset$.

It is easy to show that $k_j \leq k_j^{'}$ for each $1 \leq j \leq r$. In particular, consider the following greedy algorithm. Beginning with $T_0 = \emptyset$ until $T_p = Cj$, in each iteration p, extend T_{p-1} as by adding a coordinate $\mathbf{c}_p \in C_j \setminus T_{p-1}$ and all its repair group coordinates $\mathbf{c}_{R(p)}$ to T_{p-1}. Specifically, $T_p = T_{p-1} \cup (\mathbf{c}_{\Gamma(p)} \setminus T_{p-1})$. Now, in each iteration there must be at least one linear dependency between T_{p-1} and $\mathbf{c}_{\Gamma(p)} \setminus T_{p-1}$. Further, in each iteration, we extend the size of T by at most j, and thus, the number of iterations are at least $\left\lceil\frac{n_j}{j+1}\right\rceil$. Therefore, the number of linear dependencies among the coordinates in C_j must be at least $\left\lceil\frac{n_j}{j+1}\right\rceil$.

Case 1: Suppose we have $\mathsf{Rank}\left(S_{i-1} \cup \mathbf{c}_{\Gamma(i)}\right) \leq k-1$ throughout. Let m be the locality of the last symbol picked by the algorithm. For $1 \leq j \leq m-1$, the algorithm collects all the coordinates of locality j. Let $\hat{n}_m \leq n_m$ be the number of coordinates of locality m that are collected by the algorithm. Then, we have

$$|S| = n_1 + \cdots + n_{m-1} + \hat{n}_m.$$

Note that $\mathsf{Rank}\,(S)$ when S has accumulated all the coordinates of locality up to $m-1$ is $\mathsf{Rank}\left(\cup_{j=1}^{m-1} C_j\right) = \sum_{j=1}^{m-1} k_j$. Therefore, the rank accumulated from locality m coordinates is $(k-1) - \sum_{j=1}^{m-1} k_j := \hat{k}_m$. Now, using standard arguments similar to the proof of Theorem 8, it is easy to show that $\hat{n}_m \geq \hat{k}_m + \left\lceil \frac{\hat{k}_m}{m} \right\rceil$. Therefore,

$$|S| \geq \sum_{j=1}^{m-1} n_j + \hat{k}_m + \left\lceil \frac{\hat{k}_m}{m} \right\rceil := |S|_{LB}. \tag{4.18}$$

Next, we show that $|S|_{LB}$ is minimized when $k_j = k_j^{'}$. Let $S^{'}$ be the set collected if $\mathsf{Rank}\left(\cup_{i=0}^{j} C_i\right) - \mathsf{Rank}\left(\cup_{i=0}^{j-1} C_i\right) = k_j^{'}$. In this case the locality of the last coordinate must be r provided $\sum_{j=1}^{r-1} k_j^{'} < k-1$. Let $\hat{n}_r^{\prime}$ be the number of coordinates of locality r that are collected by the algorithm. (If $\sum_{j=1}^{r-1} k_j^{'} = k-1$, then $\hat{n}_r^{\prime} = 0$ and the following analysis still holds.) Then, we have

$$|S^{'}| = n_1 + \ldots + n_{r-1} + \hat{n}_r.$$

The rank accumulated in locality r coordinates is $(k-1) - \sum_{j=1}^{r-1} k_j^{'} := \hat{k}_r^{\prime}$. Again, using standard arguments similar to the proof of Theorem 8, it is easy to show that $\hat{n}_r^{\prime} \geq \hat{k}_r^{\prime} + \left\lceil \frac{\hat{k}_r^{\prime}}{r} \right\rceil$. Therefore,

$$|S^{'}| \geq \sum_{j=1}^{r-1} n_j + \hat{k}_r^{\prime} + \left\lceil \frac{\hat{k}_r^{\prime}}{r} \right\rceil := |S^{'}|_{LB}. \tag{4.19}$$

Next, we show that $|S^{'}|_{LB} \leq |S|_{LB}$. Suppose, for contradiction, $|S^{'}|_{LB} > |S|_{LB}$. First, note that since $k_j^{'} \geq kjj$ for $1 \leq j \leq r$, we have $r \leq m$.

Case (1a): $m = r$. Then, we have

$$\sum_{j=1}^{r-1} n_j + \hat{k}_r^{\prime} + \left\lceil \frac{\hat{k}_r^{\prime}}{r} \right\rceil > \sum_{j=1}^{r-1} n_j + \hat{k}_r + \left\lceil \frac{\hat{k}_r}{r} \right\rceil.$$

However, this essentially implies $\sum_{j=1}^{r-1} k_j^{'} < \sum_{j=1}^{r-1} k_j$, which is a contradiction.

Case (1b): $m < r$. Then, we have

$$\sum_{j=1}^{r-1} n_j + \hat{k}_r' + \left\lceil \frac{\hat{k}_r'}{r} \right\rceil > \sum_{j=1}^{r-1} n_j + n_r + \cdots + \hat{k}_r + \left\lceil \frac{\hat{k}_r}{r} \right\rceil .$$

However, this implies $\hat{k}_r' + \left\lceil \frac{\hat{k}_r'}{r} \right\rceil > n_r + \cdots + \hat{k}_r + \left\lceil \frac{\hat{k}_r}{r} \right\rceil$, which is a contradiction as $\hat{k}_r' + \left\lceil \frac{\hat{k}_r'}{r} \right\rceil \leq \hat{n}_r' \leq n_r$.

Hence, to get smallest lower bound on $|S|$, one can assign maximum incremental rank $k_j^{'}$ to each locality j. Let l_j be the number of iterations during which Algorithm 4 collects coordinates of locality j. Then, using the same arguments as in the proof of Theorem 8, we have $|S| \geq k - 1 + \sum_{j=1}^{r} l_j$ (see (4.9)). For $1 \leq j \leq r-1$, the algorithm collects all the n_j coordinates of locality j. When a coordinate of locality j is picked, the size of S can be increased by at most $j+1$ in that iteration. Thus, $l_j \geq \left\lceil \frac{n_j}{j+1} \right\rceil$ for $1 \leq j \leq r-1$. For locality r, we increment the rank of S by $(k-1) - \sum_{j=1}^{r-1} k_j^{'}$. At each step, tank is increased by at most r, thus $l_r \geq \left\lceil \frac{(k-1) - \sum_{j=1}^{r-1} k_j^{'}}{r} \right\rceil$. Hence,

$$|S| \geq k - 1 + \sum_{j=1}^{r-1} \left\lceil \frac{n_j}{j+1} \right\rceil + \left\lceil \frac{(k-1) - \sum_{j=1}^{r-1} k_j^{'}}{r} \right\rceil \geq k - 2 + \sum_{j=1}^{r-1} \left\lceil \frac{n_j}{j+1} \right\rceil + \left\lceil \frac{k - \sum_{j=1}^{r-1} k_j^{'}}{r} \right\rceil .$$

Case 2: In the last step, we get $\textsf{Rank}\left(S_{l-1} \cup \mathbf{c}_{\Gamma(l)}\right) = k$. Analysis to show that the smallest lower bound on $|S|$ is obtained assigning maximum incremental rank $k_j^{'}$ to each locality $j$is similar to Case 1.

Using the same arguments as in the proof of Theorem 8, we have $|S| \geq k - 2 + \sum_{j=1}^{r} l_j$ (see (4.14)). Following the same argument as Case 1, $l_j \geq \left\lceil \frac{n_j}{j+1} \right\rceil$ for $1 \leq j \leq r-1$. For locality r, we increment the rank of S by $k - \sum_{j=1}^{r-1} k_j^{'}$. At each step, tank is increased by at most r, thus $l_r \geq \left\lceil \frac{k-\sum_{j=1}^{r-1} k_j'}{r} \right\rceil$. Hence,

$$|S| \geq k - 2 + \sum_{j=1}^{r-1} \left\lceil \frac{n_j}{j+1} \right\rceil + \left\lceil \frac{k - \sum_{j=1}^{r-1} k_j'}{r} \right\rceil .$$

Finally, the result follows from using the Fact 1. □

4.4.2 Code Construction Based on Rank-Metric Codes

We adapt the rank-metric codes based LRC construction in [24] for the unequal all symbol locality scenario. The idea is to first precode the information symbols with a rank-metric code (in particular, with Gabidulin codes), and then use maximum distance separable (MDS) codes to obtain local parities. We begin with a brief review of rank-metric codes.

Rank-Metric Codes

Let $\mathbb{F}_q^{N \times m}$ be the set of all $N \times m$ matrices over $\mathbb{F}_q$. The *rank distance* is a distance measure between elements A and B of $\mathbb{F}_q^{N \times m}$ defined as $d_R(A)B = \mathsf{Rank}(A - B)$. It can be shown that the rank distance is indeed a metric [44]. A rank-metric code is a non-empty subset of $\mathbb{F}_q^{N \times m}$ under the context of the rank metric.

Typically, the rank-metric codes are considered by leveraging the correspondence between $\mathbb{F}_q^{1 \times m}$ and an extension field $\mathbb{F}_{q^m}$. By fixing a basis for $\mathbb{F}_{q^m}$ as an m-dimensional vector space over $\mathbb{F}_q$, any element of $\mathbb{F}_{q^m}$ can be represented as an m-length vector over $\mathbb{F}_q$. Similarly, any N-length vector over $\mathbb{F}_{q^m}$ can be represented as an $N \times m$ matrix over $\mathbb{F}_q$. The rank of a vector $A \in \mathbb{F}_{q^m}^N$ is the rank of A as an $N \times m$ matrix over $\mathbb{F}_q$, which also works for the rank distance. This correspondence allows us to view a rank-metric code in $\mathbb{F}_q^{N \times m}$ as a block code of length N over $\mathbb{F}_{q^m}$.

Focussing on linear codes, an (N, K, D) rank-metric code $\mathcal{C} \subseteq \mathbb{F}_{q^m}^N$ is a linear block code over $\mathbb{F}_{q^m}$ of length N, dimension K, and minimum rank distance D. For such codes, the Singleton bound becomes $d \leq \min\left\{1, \frac{m}{N}\right\}(N-K)+1$ (see [44]). Codes that achieve this bound are called as maximum-rank distance (MRD) codes. Note that, for $m \geq N$, the Singleton bound for rank metric coincides with the classical Singleton bound for the Hamming metric. Indeed, when $m \geq N$, every MRD code is also MDS, and hence can correct any $d-1$ erasures. We call the erasures in a rank-metric code as *rank erasures*.

Gabidulin Codes: For $N \geq m$, a class of MRD codes was presented in [44] by Gabidulin. A Gabidulin code can be obtained by evaluation of *linearized polynomials*. A linearized polynomial $f(x)$ over $\mathbb{F}_{q^m}$ of q-degree K has the form $f(x) = \sum_{i=0}^{K} a_i x^{q^i}$, where $a_i \in \mathbb{F}_{q^m}$ such that $a_K \neq 0$. Evaluation of a linearized polynomial is an $\mathbb{F}_q$-linear transform from $\mathbb{F}_{q^m}$ to itself. In other words, for any $a, b \in \mathbb{F}_q$ and $x, y \in \mathbb{F}_{q^m}$, we have $f(ax+by) = af(x)+bf(y)$.

For $m \geq N$, a codeword in an $[N,K,D]_{q^m}$ Gabidulin code $\mathcal{C}_G$ is defined as $\mathbf{c}_{\text{Gab}} = (f(g_1), \ldots, f(g_N)) \in \mathbb{F}_{q^m}^N$, where $f(x)$ is a linearized polynomial over $\mathbb{F}_{q^m}$ of q-degree $K-1$ whose coefficients are information symbols, and evaluation points $g_1, \ldots, g_N \in \mathbb{F}_{q^m}$ are linearly independent over $\mathbb{F}_q$. Note that since Gabidulin code is also an MDS code, it can correct any $N-K$ erasures [44].

Proposed Code Construction

For the simplicity of presentation, we assume that $j+1 \mid n_j$ for each j. One can generalize the construction for the case when this is not the case.

Construction 1. Consider a length-k vector of information symbols $\mathbf{m} \in \mathbb{F}_{q^m}^k$. First, we precode $\mathbf{m}$ using a Gabidulin code. Then, the codeword of the Gabidulin code is partitioned into local groups, and the local parities are computed for each group using MDS codes over $\mathbb{F}_q$. The details are as follows.

Define $N_j = n_j\left(\frac{j}{j+1}\right)$ for each $j \in [r_a]$. Let $N = \sum_{j=1}^{r_a} N_j$. First, precode the k

information symbols $\mathbf{m} \in \mathbb{F}_{q^m}^{k}$ using an $[N, k, N{-}k{+}1]_{q^m}$ Gabidulin code to obtain $\mathbf{c}_{\text{Gab}} \in \mathbb{F}_{q^m}^{N}$. Partition $\mathbf{c}_{\text{Gab}}$ into r_a disjoint groups $\mathbf{c}_{\text{Gab}}^{j}$ each of size N_j, *i.e.*, $\mathbf{c}_{\text{Gab}} = \cup_{j=1}^{r_a} \mathbf{c}_{\text{Gab}}^{j}$, with $\mathbf{c}_{\text{Gab}}^{j} = \emptyset$ for each j such that $N_j = 0$. For each $1 \leq j \leq r_a$ such that $N_j > 0$, further partition $\mathbf{c}_{\text{Gab}}^{j}$ symbols into $\frac{N_j}{j}$ disjoint local groups $\mathbf{c}_{\text{Gab}}^{j,i}$ each of size j, *i.e.*, $\mathbf{c}_{\text{Gab}}^{j} = \cup_{i=1}^{\frac{N_j}{j}} \mathbf{c}_{\text{Gab}}^{j,i}$.

For each group $\mathbf{c}_{\text{Gab}}^{j,i}$ of j symbols, generate a local parity using a $(j+1, j, 2)$ MDS code over $\mathbb{F}_q$. Denote the resulting code as $\mathcal{C}_{\text{LRC}}$. Note that the total number of symbols are $\sum_{j=1}^{r_a} \frac{N_j}{j}(j+1) = \sum_{j=1}^{r_a} n_j = n$. Moreover, note that, the partitioning for obtaining local parities is performed in such a way that $\mathcal{C}_{\text{LRC}}$ possesses all-symbol locality profile $(n_1, \ldots, n_{r_a})$. Next, we show that the above construction achieves the distance bound mentioned in Theorem 9. Next, we show that the above construction achieves the distance bound mentioned in Theorem 9.

Theorem 10. *Let $\mathcal{C}_{LRC}$ be an (n, k, d) LRC with all-symbol locality profile $(n_1, \ldots, n_{r_a})$ obtained by Construction 1. If $j + 1 \mid n_j$ for each $j \in [r_a]$, then $\mathcal{C}_{LRC}$ over $\mathbb{F}_{q^m}$ such that $m \geq \sum_{j=1}^{r_a} n_j \left(\frac{j}{j+1}\right)$ and $q \geq r_a + 1$, attains the minimum distance bound in* (4.17).

Proof. Let $e := n - k + 1 - \sum_{j=1}^{r-1} \left\lceil \frac{n_j}{j+1} \right\rceil - \left\lceil \frac{k - \sum_{i=1}^{r-1}\left(n_j - \left\lceil \frac{n_j}{j+1} \right\rceil\right)}{r} \right\rceil$. Similar to [24], the idea is show that any e symbol erasures correspond to $N - K$ rank erasures, which can be corrected by the Gabidulin code.

The $\mathbb{F}_q$-linearity of the linearized polynomials plays a crucial role. In particular, since the local parities are obtained using an MDS code over $\mathbb{F}_q$, any symbol $\mathbf{c}_i$ of locality j can be written as $\mathbf{c}_i = \sum_{p=1}^{j} a_p \mathbf{c}_{i_p} = \sum_{p=1}^{j} a_p f(g_{i_p}) = f\left(\sum_{p=1}^{j} a_p g_{i_p}\right)$. Hence, for each $j \in [r_a]$, in a local group of size j, any $m \leq j$ symbols are evaluations of $f(x)$ in m points that are linearly independent over $\mathbb{F}_q$. Therefore, for each $j \in [r_a]$, in a local group of size $j+1$, any $i+1 (\leq j+1)$ symbol erasures correspond to i rank erasures. Moreover, taking any j points from all local groups of size $j+1$ for each $j \in [r_a]$, we obtain the Gabidulin codeword, which has obtained by precoding $\mathbf{m}$.

With above observation, the worst case erasure pattern is when the erasures occur in the smallest possible number of local groups (of possibly different localities), and the number of erasures in each local group are maximal.

Note that we can write n as $n = \sum_{j=1}^{r_a} N_j + \frac{N_j}{j}$. Let $k = \sum_{j=1}^{r-1} N_j + N_r^{'}$ for some $N_r^{'} < N_r$. Then, we can write

$$e = 1 + \sum_{j=r}^{r_a} \left(N_j + \frac{N_j}{j} \right) - \left(N_r^{'} + \left\lceil \frac{N_r^{'}}{r} \right\rceil \right). \tag{4.20}$$

On the other hand, for the outer Gabidulin code, we have

$$N - k = \sum_{j=r}^{r_a} N_j - N_r^{'}. \tag{4.21}$$

Case 1: $r \mid N_r^{'}$. Let $N_r^{'} = r\beta$. Then, from (4.20), we have $e = 1 + \sum_{j=r+1}^{r_a} (j+1) \left(\frac{n_j}{j+1} \right) + (r+1) \left(\frac{n_r}{r+1} - \beta \right)$. Thus, in the worst case, the number of local groups that are completely erased are $\sum_{j=r+1}^{r_a} \left(\frac{n_j}{j+1} \right) + \left(\frac{n_r}{r+1} - \beta \right)$ with one erasure in an additional group. Recall that, due to the $\mathbb{F}_q$-linearity, any $i+1$ erasures in a local group of size $j+1$, the number of rank erasures corresponding to the Gabidulin codeword are only j. Thus, total number of rank erasures are $\sum_{j=r+1}^{r_a} j \left(\frac{n_j}{j+1} \right) + r \left(\frac{n_r}{r+1} - \beta \right)$.

However, from (4.21), we get $N - k = \sum_{j=r+1}^{r_a} j \left(\frac{n_j}{j+1} \right) + r \left(\frac{n_r}{r+1} - \beta \right)$. Therefore, all the rank erasures can be corrected by the outer Gabidulin code.

Case 2: $r \nmid N_r^{'}$. Let $N_r^{'} = r\beta + \gamma$, where $1 \leq \gamma \leq r-1$. Then, from (4.20), we have $e = 1 + \sum_{j=r+1}^{r_a} (j+1) \left(\frac{n_j}{j+1} \right) + (r+1) \left(\frac{n_r}{r+1} - \beta - 1 \right) + (r - \gamma + 1)$. In other words, in the in the worst case, the number of local groups that are completely erased are $\sum_{j=r+1}^{r_a} \left(\frac{n_j}{j+1} \right) + \left(\frac{n_r}{r+1} - \beta - 1 \right)$ with $(r - \gamma + 1)$ erasures in an additional group. This corresponds to $\sum_{j=r+1}^{r_a} j \left(\frac{n_j}{j+1} \right) + r \left(\frac{n_r}{r+1} - \beta - 1 \right) + (r - \gamma)$ rank erasures.

From (4.21), we get $N - k = \sum_{j=r+1}^{r_a} j \left(\frac{n_j}{j+1}\right) + r\left(\frac{n_r}{r+1} - \beta - 1\right) + (r - \gamma)$. Hence, all the rank erasures can be corrected by the outer Gabidulin code. □

4.5 Information Locality Requirement

Consider a code design scenario, wherein each information symbol has an upper limit on its locality. For instance, suppose we need to design a code of dimension 6, in which, the first 3 information symbols should have locality at most 2, and the remaining 3 information symbols should have locality at most 3. For this case, we say that we have *information locality requirement* of $\tilde{\mathbf{k}} = (0, 3, 3)$. Note that this requirement is equivalent to the condition that the code should contain at least 3 information symbols of locality up to 2, and at least 6 information symbols of locality up to 3. We use this later condition to formally define the notion of locality requirement as follows.

Definition 9. *Let $\tilde{\mathbf{k}} = (\tilde{k}_1, \ldots, \tilde{k}_r)$ be a length-r vector for some $r < k$ such that for each $1 \leq j \leq r$, we have $0 \leq \tilde{k}_j \leq k$ and $\sum_{j=1}^{r} \tilde{k}_j = k$. Consider a code $\mathcal{C}$ with information locality profile $\mathbf{k} = (k_1, \ldots, k_{r'})$ for some $r' \leq r$. We say that $\mathcal{C}$ satisfies the information locality requirement $\tilde{\mathbf{k}}$, if, for each $1 \leq i \leq r$, we have $\sum_{j=1}^{i} k_j \geq \sum_{j=1}^{i} \tilde{k}_j$, where we set $k_j = 0$ for $r' + 1 \leq j \leq r$ if $r' < r$. In this case, we say that the locality profile $\mathbf{k}$ satisfies the locality requirement $\tilde{\mathbf{k}}$, which is denoted as $\mathbf{k} \succeq \tilde{\mathbf{k}}$.*

A number of locality profiles can satisfy the given locality requirement $\tilde{\mathbf{k}}$. For example, one can find a number of locality profiles that satisfy $\tilde{\mathbf{k}} = (0, 3, 3)$, such as $\mathbf{k}_1 = (2, 4, 0)$, $\mathbf{k}_2 = (3, 0, 3)$, $\mathbf{k}_3 = (0, 6, 0)$, $\mathbf{k}_4 = (1, 2, 3)$; while, for instance, $\mathbf{k}_5 = (2, 0, 4)$ does not satisfy $\tilde{\mathbf{k}}$. Observe that, among those profiles satisfying $\tilde{\mathbf{k}}$, the last two locality profiles result in the largest minimum distance (see (4.4)). In general, a large number of locality profiles can satisfy a given locality requirement. We are interested to find one of these locality profiles that maximizes the minimum distance bound given by (4.4). We call such an information locality profile as an *optimal* locality profile. We present Algorithm 5 that greedily finds an optimal locality profile for a given locality requirement. This allows us to use Pyramid code construction to design codes that have maximum minimum distance among all codes that satisfy the locality requirement

Give a locality requirement $\tilde{\mathbf{k}}$, we are interested in finding a locality profile $\mathbf{k} \succeq \tilde{\mathbf{k}}$ which results in largest upper bound on the minimum distance for fixed n. More formally, we can define the problem as follows.

$$\min_{\mathbf{k}\in\mathbb{Z}_+^r} \quad \sum_{j=1}^{r} \left\lceil \frac{k_j}{j} \right\rceil \qquad\qquad (P1) \tag{4.22}$$

$$\text{s.t.} \quad \sum_{j=1}^{i} k_j \geq \sum_{j=1}^{i} \tilde{k}_j, \ \text{ for } 1 \leq i \leq r, \tag{4.23}$$

$$\text{and} \quad \sum_{j=1}^{r} k_j = \sum_{j=1}^{r} \tilde{k}_j \qquad\qquad . \tag{4.24}$$

A solution of the above optimization problem is said to be an *optimal* locality profile. In the following we give a greedy algorithm which finds an optimal $\mathbf{k}^*$. From $\sum_{j=1}^{r-1} k_j \geq \sum_{j=1}^{r-1} \tilde{k}_j$ and $\sum_{j=1}^{r} k_j = \sum_{j=1}^{r} \tilde{k}_j$, we get that $k_r \leq \tilde{k}_r$. In similar way, we can see that the inequality constraints above can be replaced by $\sum_{j=i}^{r} k_j \geq \sum_{j=i}^{r} \tilde{k}_j$ and $\sum_{j=1}^{r} k_j = \sum_{j=1}^{r} \tilde{k}_j$. The idea of the algorithm is to start with the largest locality r and set k_r^* as the largest multiple of r such that $k_r^* \leq \tilde{k}_r$. Move the residue $\tilde{k}_r - k_r^*$ to the next locality $r-1$, and set k_{r-1}^* as the largest multiple of r such that $k_{r-1}^* \leq \tilde{k}_{r-1} + \tilde{k}_r - k_r^*$. We continue this until we reach locality 1.

Remark 19. *Note that Algorithm 5 assigns $k_j^* = \tilde{k}_j + \gamma_{j+1} - \gamma_j$ for each locality j. This gives $\sum_{j=i}^{r} k_j^* = \sum_{j=i}^{r} \tilde{k}_j - \gamma_i$ for each $r \geq i \geq 1$.*

Theorem 11. *For a given information locality requirement $\tilde{\mathbf{k}}$, let $\mathbf{k}^*$ be the output of Algorithm 5. Then, $\mathbf{k}^* \succeq \tilde{\mathbf{k}}$, and $\mathbf{k}^*$ results in the largest value of the minimum distance bound*

named 5 Find an optimal locality profile $\mathbf{k}^*$ for a given locality requirement $\tilde{\mathbf{k}}$

```
Set $\gamma_{r+1} = 0, j = r$
while $j \geq 1$ do
    Chose integers $\beta_j$ and $\gamma_j$ such that $\tilde{k}_j + \gamma_{j+1} = j\beta_j + \gamma_j$
    Set $k_j^* = j\beta_j$
    Decrement $j$
end while
```

in (4.4) *among all* $\mathbf{k}$ *such that* $\mathbf{k} \succeq \tilde{\mathbf{k}}$.

Proof. The idea is to show that any optimal information locality profile can be transformed into a form of $\mathbf{k}^*$ without loosing optimality. We first prove that it is always possible to obtain an optimal information locality profile $\mathbf{k}^{''}$ such that $j \mid k_j^{''}$ for each $j \in [r]$.

Lemma 3. *Given a locality requirement* $\tilde{\mathbf{k}}$*, any optimal information locality profile* $\mathbf{k}^{'}$ *can be converted into another optimal information locality profile* $\mathbf{k}^{''}$ *such that* $j \mid k_j^{''}$ $\forall j \in [r]$.

Proof. By induction on the number of localities j such that $j \nmid k_j^{'}$. Let $|\{j : j \nmid k_j^{'}\}| = m$.

Basis step: $m = 1$. Let $j_m \in [r]$ be the only locality such that $j_m \nmid k_{j_m}^{'}$. We can write $k_{j_m}^{'} = j_m\beta_{j_m} + \gamma_{j_m}$ such that $1 \leq \beta_{j_m} \leq j_m - 1$. Set $k_{j_m}^{''} = k_{j_m}^{'} - \gamma_{j_m}$, $k_{\gamma_{j_m}}^{''} = k_{\gamma_{j_m}}^{'} + \gamma_{j_m}$, and $k_j^{''} = k_j^{'}$ for all $j \in [r]$ such that $j \neq j_m$, $j \neq \gamma_{j_m}$.

First, observe that $\mathbf{k}^{''}$ is such that $j \mid k_j$ for each $j \in [r]$, since $\gamma_{j_m} \mid k_{\gamma_{j_m}}^{'}$.

Second, note that $\mathbf{k}^{''}$ is a feasible solution for (P1). This is because, for $1 \leq i \leq j_m - 1$, we have $\sum_{j=1}^{i} k_j^{''} = \sum_{j=1}^{i} k_j^{'} + \gamma_{j_m} \geq \sum_{j=1}^{i} \tilde{k}_j$, and for $j_m \leq i \leq r$, we have $\sum_{j=1}^{i} k_j^{''} = \sum_{j=1}^{i} k_j^{'} \geq \sum_{j=1}^{i} \tilde{k}_j$. For both these cases, the inequality follows since $\mathbf{k}^{'}$ satisfies the constraints of (P1).

Finally, it is easy to see that $\mathbf{k}^{''}$ is also optimal, since $\left\lceil \frac{k_{j_m}^{''}}{j_m} \right\rceil = \left\lceil \frac{k_{j_m}^{'}}{j_m} \right\rceil - 1$, $\left\lceil \frac{k_{\gamma_{j_m}}^{''}}{\gamma_{j_m}} \right\rceil = \left\lceil \frac{k_{\gamma_{j_m}}^{'}}{\gamma_{j_m}} \right\rceil + 1$, and $\left\lceil \frac{k_j^{''}}{j} \right\rceil = \left\lceil \frac{k_j^{'}}{j} \right\rceil$ for the rest of the localities.

Induction step: $m \geq 2$. Suppose the hypothesis holds whenever $|\{j : j \nmid k_j^{'}\}| \leq m - 1$.

Consider the case when $|\{j : j \nmid k_j^{'}\}| = m$. Denote such a set of localities as $\{j_1, \ldots, j_m\}$, where $j_1 < \cdots < j_m$. Now, we can write $k_{j_m}^{'} = j_m\beta_{j_m} + \gamma_{j_m}$ such that $1 \leq \beta_{j_m} \leq j_m - 1$. Set $k_{j_m}^{'} = k_{j_m}^{'} - \gamma_{j_m}$, and $k_{\gamma_{j_m}}^{'} = k_{\gamma_{j_m}}^{'} + \gamma_{j_m}$.

Similar to $m = 1$ case, we can verify that $\mathbf{k}^{'}$ remains to be an optimal solution to (P1) after the transformation. Further, since $j_m \mid k_{j_m}^{'}$, we get $|\{j : j \nmid k_j^{'}\}| = m - 1$. Then, the proof follows by the induction hypothesis. □

Let $|\{j : k_j^{''} \neq k_j^*\}| = m$. Denote such a set of localities as $\{j_1, \ldots, j_m\}$, where $j_1 < \cdots < j_m$. We first prove some properties for the localities where the coordinate values differ.

Proposition 2. $k_{j_m}^{''} < k_{j_m}^*$

Proof. Suppose, for contradiction, $k_{j_m}^{''} > k_{j_m}^*$. We can write $k_{j_m}^{''} = k_{j_m}^* + pj_m$ for some integer $p \geq 1$, since both $k_{j_m}^{''}$ and $k_{j_m}^*$ are multiples of j_m. Consider

$$\sum_{i=j_m}^{r} k_i^{''} = k_{j_m}^* + pj_m + \sum_{i=j_m+1}^{r} k_i^* \tag{4.25}$$

$$= \sum_{i=j_m}^{r} \tilde{k}_i - \gamma_{j_m} + pj_m \tag{4.26}$$

$$\geq \sum_{i=j_m}^{r} \tilde{k}_i - (j_m - 1) + pj_m \tag{4.27}$$

$$\geq \sum_{i=j_m}^{r} \tilde{k}_i + (p-1)j_m + 1 \tag{4.28}$$

$$\geq \sum_{i=j_m}^{r} \tilde{k}_i. \tag{4.29}$$

However, this contradicts the feasibility of $\mathbf{k}^{''}$ as it should satisfy $\sum_{j_m}^{r} k_i^{''} \leq \sum_{j_m}^{r} \tilde{k}_i$ (due to $\sum_{i=1}^{j_m} k_i^{''} \geq \sum_{i=1}^{j_m} \tilde{k}_i$ and $\sum_{i=1}^{r} k_i^{''} = \sum_{i=1}^{r} \tilde{k}_i$). □

Next, we show that for any information locality profile, moving the coordinates to the higher locality does not increase the minimum distance bound.

Proposition 3. *Consider an information locality profile* $\mathbf{k}$*. For any locality pair* i *and* j *such that* $i < j$ *and* $k_j > 0$*. Set* $k_i = k_i - \delta$ *and* $k_j = k_j + \delta$ *for an integer* δ *such that either* $i \mid \delta$ *or* $j \mid \delta$ *(or both). Then, such a transformation does not increase the value of the minimum distance bound.*

Proof. Case 1: $i \mid \delta$. Let $\delta = ia$ for some integer a. After moving the coordinates of locality i to locality j, the term $\left\lceil \frac{k_i}{i} \right\rceil$ reduces by a. Whereas, the term $\left\lceil \frac{k_j}{j} \right\rceil$ increases by at most $\left\lceil \frac{\delta}{j} \right\rceil$, which itself is at most a.

Case 2: $j \mid \delta$. Let $\delta = jb$ for some integer b. In this case, the term $\left\lceil \frac{k_j}{j} \right\rceil$ increases by b. Whereas, the term $\left\lceil \frac{k_i}{i} \right\rceil$ reduces by at least $\left\lfloor \frac{\delta}{i} \right\rfloor$, which itself is at least b.

Therefore, in both the above case, the value of (4.22) does not increase. □

Finally, we show that for any information locality profile, moving the coordinates to the lower locality to obtain divisibility does not change the minimum distance bound.

Proposition 4. *Consider an information locality profile* $\mathbf{k}$*. Let* j *be a locality such that* $j \nmid k_j$*, and let* $k_j = j\beta_j + \gamma_j$ *for some integers* β_j *and* $1 \leq \gamma_j \leq j - 1$*. Then, setting* $k_j = k_i - \gamma_j$ *and* $k_{\gamma_j} = k_{\gamma_j} + \gamma_j$ *does not change the value of the minimum distance bound.*

Proof. The argument is the same as for the basis step in the proof of Lemma 3. □

Finally, we show that we can transform an optimal information locality profile where divisibility holds for each locality into $\mathbf{k}^*$.

Lemma 4. *Given a locality requirement* $\tilde{\mathbf{k}}$*, any optimal information locality profile* $\mathbf{k}^{'}$*, where* $j \mid k_j^{''}$ *for each* j*, can be converted into* $\mathbf{k}^*$ *without loosing optimality, where* $\mathbf{k}^*$ *is the output of Algorithm 5.*

named 6 Transform an optimal locality profile $\mathbf{k}^{''}$ to $\mathbf{k}^{*}$

```
 1: Let $|\{j : k^{''}_j \neq k^*_j\}| = m$
 2: **while** $m > 0$ **do**
 3:     Let $j_m = \max\{j : k^{''}_j \neq k^*_j\}$
 4:     **while** $k^{''}_{j_m} < k^*_{j_m}$ **do**
 5:         Let $j_p = \max\{j : k^{''}_j > k^*_j\}$
 6:         Let $\delta_{j_m} = k^*_{j_m} - k^{''}_{j_m}$, $\delta_{j_p} = k^{''}_{j_p} - k^*_{j_p}$
 7:         Set $k^{''}_{j_m} = k^{''}_{j_m} + \min\{\delta_{j_m}, \delta_{j_p}\}$, $k^{''}_{j_p} = k^{''}_{j_p} - \min\{\delta_{j_m}, \delta_{j_p}\}$
 8:         **if** $\delta_{j_m} < \delta_{j_p}$ **then**
 9:             Let $k^{''}_{j_p} = j_p\beta_{j_p} + \gamma_{j_p}$
10:             **if** $\gamma_{j_p} > 0$ **then**
11:                 Set $k^{''}_{j_p} = k^{''}_{j_p} - \gamma_{j_p}$, $k^{''}_{\gamma_{j_p}} = k^{''}_{\gamma_{j_p}} + \gamma_{j_p}$
12:             **end if**
13:         **end if**
14:     **end while**
15:     Set $m = |\{j : k^{''}_j \neq k^*_j\}|$
16: **end while**
```

Proof. We give an iterative algorithm (Algorithm 6) to transform an optimal information locality profile $\mathbf{k}^{''}$ to $\mathbf{k}^*$. First note that, by Proposition 2, it must be that $k^{''}_{j_m} < k^*_{j_m}$ in the first iteration of the outer while-loop. Moreover, at line 13, $\mathbf{k}^{''}$ is such that $j \mid k^{''}_j$ for each $j \in [r]$, hence we can invoke Proposition 2 for the every iteration of outer while-loop. Next, the optimality of $\mathbf{k}^{''}$ is maintained at line 6 due to Proposition 3, and also at line 10 due to Proposition 4. Finally, Algorithm 6 must terminate in finite time as m decreases by at least 1 at line 13. □

The proof of Theorem 11 follows from Lemma 3 and Lemma 4

□

4.6 Conclusion

We investigate linear codes, in which, different nodes possess different values of locality. We first considered a class of codes with *unequal information locality*, *i.e.*, systematic codes with unequal locality constraints imposed only on the information symbols. We computed a tight upper bound on the minimum distance as a function of locality constraints. We demonstrated that the construction of Pyramid codes by Huang *et al.* [18] can be adapted to design *optimal* codes with unequal information locality that achieve the minimum distance bound. Then, we considered codes with *unequal all-symbol locality*, *i.e.*, codes in which the locality constraints are imposed on all symbols. We established a tight upper bound on the minimum distance as a function of number of symbols of each locality value. We showed that the construction based on rank-metric codes by Silberstein *et al.* [24] can be adapted to obtain optimal codes with unequal all-symbol locality. Finally, we introduced the concept of *locality requirement* of a code, which can be viewed as a recoverability requirement on symbols. Information locality requirement of a code essentially specifies the minimum number of information symbols of each locality value that must be present in the code. For a given locality requirement, we present a greedy algorithm to construct codes that have maximum minimum distance among all codes that satisfy the locality requirement.

5. CODES WITH LOCALITY IN THE RANK AND SUBSPACE METRICS*

5.1 Introduction

Distributed storage systems have been traditionally replicating data over multiple nodes to guarantee reliability against failures and protect the data from being lost [76, 77]. However, the enormous growth of data being stored or computed online has motivated practical systems to employ erasure codes for handling failures (*e.g.*, [7, 78]). This has galvanized a significant amount of work in the past few years on novel erasure codes that efficiently handle node failures in distributed storage systems. Two main families of codes have received primary research attention: (a) *regenerating codes* – that minimize repair bandwidth, *i.e.*, the amount of data downloaded while repairing a failed node (see, *e.g.*, [8, 57, 13]); and (b) *locally repairable codes* – that minimize locality, *i.e.*, the number of nodes participating in the repair process (see, *e.g.*, [18, 9, 20, 79, 25]). Almost all the work in the literature on these families has considered block codes under the Hamming metric.

In this work, we first focus our attention to codes with locality constraints in the rank metric. Let $\mathbb{F}_q$ be the finite field of size q. Codewords of a rank-metric code are $m \times n$ matrices over $\mathbb{F}_q$, where the rank distance between two matrices is the rank of their difference [80, 44, 81]. Maximum rank distance (MRD) codes are analogues of the maximum distance separable (MDS) codes in the Hamming metric. We are interested in rank-metric codes with locality constraints. To quantify the requirement of locally under the rank metric, we introduce the notion of *rank-locality*. We say that the i-th column of an $m \times n$ array code has (r, δ) *rank-locality* if there exists a set $\Gamma(i)$ of $r + \delta - 1$ columns containing i such that the array code formed by deleting the columns outside $\Gamma(i)$ has rank distance at

*Parts of this chapter are reprinted with permission from [75] "Rank-metric codes with local recoverability," by S. Kadhe, S. El Rouayheb, I. Duursma, and A. Sprintson, 2016. *In Proceedings of 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1033-1040, Sept 2016. Copyright © by IEEE.

least δ. We say that an $m \times n$ array code has (r, δ) rank-locality if every column has (r, δ) rank-locality.

Our motivation of considering rank-locality is to design codes that can locally recover from *rank errors and erasures*. Rank-errors are the error patterns such that the rank of the error matrix is limited. For instance, consider an error pattern corrupting a 4×4 bit array shown in Fig. 5.1. Though this pattern corrupts half the bits, its rank over the binary field is only one. Note that it is not possible to correct such an error pattern using a code equipped with the Hamming metric. On the other hand, rank-metric codes are well known for their ability to effectively correct rank-errors [81, 82].

$$E \;=\; \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Figure 5.1: A rank-error pattern of rank one.

Errors and erasures that affect a limited number of rows and/or columns are usually referred to as *crisscross patterns* [81, 82]. (See Fig. 5.2 for some examples of crisscross erasures.) Our goal is to investigate codes that can *locally* recover from crisscross erasures (and rank-errors). We note that crisscross errors (with no locality) have been studied previously in the literature [81, 82], motivated by applications in memory chip arrays and multi-track magnetic tapes. Our renewed interest in these types of failures stems from the fact that they form a subclass of *correlated and mixed failures*.

Recent research has shown that many distributed storage systems suffer from a large number of correlated and mixed failures [4, 83, 84, 85, 86, 87]. For instance, correlated failure of several nodes can occur due to, say, simultaneous upgrade of a group of servers, or a failure of a rack switch or a power supply shared by several nodes [4, 83, 84]. Moreover, in distributed storage systems composed of solid state drives (SSDs), it is not uncommon to have a failed SSD along with a few corrupted *blocks* in the remaining SSDs, referred to as mixed failures [87, 88, 89]. Therefore, recent research on coding for distributed storage has also started focusing on correlated and/or mixed failure models, see *e.g.*, [90, 91, 92, 69, 93, 88, 94, 95].

Another potential application for codes with rank-locality is for correcting errors occurring in dynamic random-access memories (DRAMs). In particular, a typical DRAM chip contains several internal *banks*, each of which is logically organized into *rows* and *columns*. Each row/column address pair identifies a *word* composed of several bits. Recent studies show that DRAMs suffer from non-negligible percentage of bit errors, single-row errors, single-column errors and single-bank errors [96, 97, 98]. Using an array code across banks, with a local code for each bank can be helpful in correcting such error patterns.

In general, our goal is to design and analyze codes that can *locally recover* the crisscross patterns that affect a limited number of rows and columns by accessing a small number of nodes. We show that a code with (r, δ) rank-locality can locally repair any crisscross erasure pattern that affects less than δ rows and columns by accessing only r columns. We begin with a toy example to motivate the coding theoretic problem that we seek to solve.

Example 5. *Consider a datacenter, such as the one depicted in Fig. 5.2, consisting of multiple racks, each of which containing a number of servers. Each server is composed of a number of storage nodes which can either be solid state drives (SSDs) or hard disk drives (HDDs).*[2] *Given two positive integers δ and d such that $\delta < d$, our goal is to encode*

[2] Many practical storage systems such as Facebook's 'F4' storage system [78] and all-flash storage arrays such as [99, 100] have similar architecture.

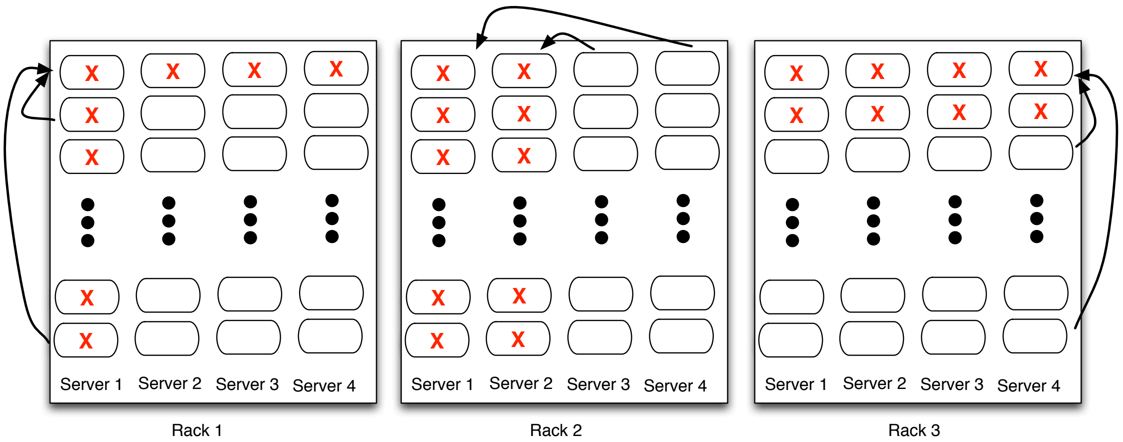


Figure 5.2: A few instances of crisscross failures affecting two rows and/or columns.

the data in such a way that

1. *any crisscross failure affecting at most δ rows and/or columns of nodes in a rack should be 'locally' recoverable by only accessing the nodes on the corresponding rack, and*

2. *any crisscross failure that affects at most d rows and/or columns of nodes in the datacenter should be recoverable (potentially by accessing all the remaining data).*

Note that the failure patterns of the first kind can occur in several cases. For example, all the nodes on a server would fail if, say, the network switch connecting the server to the system fails. The entire row of nodes would be temporarily unavailable if these nodes are simultaneously scheduled for an upgrade. A few locally recoverable crisscross patterns are shown in Fig. 5.2 (considering $\delta = 2$). Note that locally recoverable erasures in different racks can be simultaneously repaired.

Next, we extend the notion of locality from the rank metric to the subspace distance metric. Let $\mathbb{F}_q^M$ denote the vector space of M-tuples over $\mathbb{F}_q$. A subspace code is a non-empty set of subspaces of $\mathbb{F}_q^M$. A code in which each codeword has the same dimension is called a constant-dimension code. A useful distance measure between two spaces U and V, called *subspace metric*, is defined in [101] as $d_S(U, V) = \mathsf{dim}(U) + \mathsf{dim}(V) - 2\,\mathsf{dim}(U \cap V)$. Note that every codeword of a subspace code can be associated with an ordered basis. For a constant-dimension code, we say that the i-th basis vector has (r, δ) *subspace-locality* if there exists a set $\Gamma(i)$ of at most $r + \delta - 1$ basis vectors containing i such that the code obtained by removing the basis vectors outside $\Gamma(i)$ for each codeword has subspace distance at least δ. We say that a constant-dimension subspace code has (r, δ) rank-locality if every basis vector has (r, δ) rank-locality. Subspace codes play an important role in correcting errors and erasures (rank-deficiencies) in non-coherent linear network coding [101, 102]. One potential application of these novel subspace codes with locality is for downloading data and repairing failed nodes in a distributed storage system, in which the nodes are connected over a network that can introduce errors and erasures. The locality is useful in repairing a failed storage node over the network, or when a user wants to download partial data by connecting to only a small subset of nodes.

Our Contributions

First, we introduce the notion of locality in rank metric. Then, we establish a tight upper bound on the minimum rank distance of codes with (r, δ) rank-locality. We construct a family of *optimal* codes which achieve this upper bound. Our approach is inspired by [25], which generalizes Reed-Solomon code construction to obtain codes with locality. We generalize the Gabidulin code construction [44] to design codes with rank-locality. In particular, we obtain codes as evaluations of specially constructed *linearized polynomials* over an extension field, and our codes reduce to Gabidulin codes if the locality parameter r equals the code dimension. Finally, we characterize various erasure and error patterns that the proposed codes with rank-locality can efficiently correct.

Second, we extend the notion of locality to the subspace metric. Then, we consider a method to construct subspace codes by *lifting* rank-metric codes proposed in [103]. We show that a subspace code obtained by lifting an array code with rank-locality possesses subspace-locality. This enables us to construct a family of subspace codes with locality by lifting the proposed rank-metric codes with locality.

Related Work

Codes with Locality: Consider a block code of length n that encodes k information symbols. A symbol i is said to have *locality* r if it can be recovered by accessing r other symbols in the code. Note that r is the minimum possible size of a *recovering set* for the i-th symbol. We say that a code has locality r if each of its n symbols has locality at most r.

Codes with small locality were introduced in [18, 19] (see also [20]). The study of the locality property was galvanized with the pioneering work of Gopalan *et al.* [9]. One of their key contributions was to establish a trade-off between the minimum distance of a scalar linear code and its locality analogous to the classical Singleton bound.

The distance bound was generalized for codes with multiple local parities in [59], for universal (scalar/vector linear, nonlinear) codes in [21], and for universal codes with multiple parities in [22, 60]. A large number of optimal code constructions have been presented, see *e.g.*, [24, 61, 62, 25, 55, 63, 64, 26, 65, 66]. In particular, it is worth noting the following two references. In [24], the authors construct optimal LRCs using rank-metric codes as outer codes, and in [25] the authors generalize Reed-Solomon code construction to design LRCs with small alphabet size.[3]

Rank-Metric Codes: Rank-metric codes were introduced by Delsarte [80] and were largely developed by Gabidulin [44] (see also [81]). In addition, Gabidulin [44] presented

[3] We present a detailed comparison of our proposed constructions with those of [25] and [24] in Sections 6 and 6, respectively.

a construction for a class of MRD codes. Roth [81] introduced the notion of crisscross error pattern, and showed that MRD codes are powerful in correcting such error patterns. In [82], the authors presented a family of MDS array codes for correcting crisscross errors.

Codes for Mixed Failures: Several families of codes have recently been proposed to encounter mixed failures. The two main families are: sector-disk (SD) codes and partial-MDS (PMDS) codes (see [88, 94, 104, 95]). These codes consider the set up when data is stored in an $m \times n$ array, where a column of an array can be considered as an SSD. Each row of the array contains up to k data symbols and $h = n - k$ parity symbols, which together form a maximum distance separable (MDS) code. Furthermore, there are s global parity symbols in the first k columns. SD codes can tolerate erasure of any h drives, plus erasure of any additional s sectors in the array. PMDS codes can tolerate a broader class of erasures: any h sector erasures per row, plus any additional s sector erasures.

Sector-Disk (SD) codes for correcting mixed failures, *i.e.*, disk failures and sector failures, were introduced in [94]. Partial-MDS (PMDS) codes for correcting mixed failures were introduced in [88]. Since then, there have been several constructions for SD and PMDS codes, see, *e.g.*, [104, 95], and references therein.

Codes for Correlated Failures: Very recently, Gopalan *et al.* [93] presented a class of *maximally recoverable* (MR) codes for *grid-like topologies*. For an $m \times n$ array, the grid-like topology essentially specifies the number of *local* parity check equations in every row and every column, and the number of *global* parity check equations in the array. The maximal recoverability means that the code has the *strongest* erasure correction capability that is possible with a given grid-like topology. The notion of maximal recoverability was first proposed by [18] and was generalized by [92]. We note that the class of codes considered in [93] can be used to correct mixed failures as well.

Subspace Codes: The important role of the subspace metric in correcting errors and erasures in non-coherent linear network codes was first noted in [101]. Since then, subspace codes (also known as codes over projective space) have been studied in a number of research papers, see *e.g.*, [102, 103, 105, 106, 107, 108, 109, 110, 111, 112, 113], and references therein.

Codes for Distributed Storage Based on Subspace Codes: Recently, subspace codes have been used to construct repair efficient codes for distributed storage systems. In [114], the authors construct regenerating codes based on subspace codes. In [115], array codes with locality and availability (in the Hamming metric) are constructed using subspace codes. A key feature of the proposed codes is their small locality for recovering a lost symbol as well as a lost column.

5.2 Preliminaries

Notation: We use the following notation. For an integer l, $[l] = \{1, 2, \ldots, l\}$. For a vector $\mathbf{x}$, $\mathsf{wt}\left(\mathbf{x}\right)$ denotes its Hamming weight, *i.e.*, $\mathsf{wt}\left(\mathbf{x}\right) = |\{i : \mathbf{x}(i) \neq 0\}|$. The rank an the column space of a matrix H is denoted by $\mathsf{Rank}\left(H\right)$ and $\langle H\rangle$, respectively. The linear span of a set of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$ is denoted by $\langle \mathbf{x}_1, \ldots, \mathbf{x}_k\rangle$. The reduced column echelon form (RCEF) of a matrix H is denoted by $\mathsf{rcef}\left(H\right)$.

Let $\mathcal{C}$ denote a linear (n, k) code over $\mathbb{F}_q$ with block-length n, dimension k, and minimum distance $d_{min}\left(\mathcal{C}\right)$. For instance, under Hamming metric, we have $d_{min}\left(\mathcal{C}\right) = \min_{\mathbf{c}_i, \mathbf{c}_j \in \mathcal{C},\, \mathbf{c}_i \neq \mathbf{c}_j} \mathsf{wt}\left(\mathbf{c}_i - \mathbf{c}_j\right)$. Given a length-$n$ block code $\mathcal{C}$ and a set $\mathcal{S} \subset [n]$, let $\mathcal{C} \mid_{\mathcal{S}}$ denote the restriction of $\mathcal{C}$ on the coordinates in $\mathcal{S}$. Essentially, $\mathcal{C} \mid_{\mathcal{S}}$ is the code obtained by puncturing $\mathcal{C}$ on $[n] \setminus \mathcal{S}$.

Recall that, for Hamming metric, the well known Singleton bound gives an upper bound on the minimum distance of an (n, k) code $\mathcal{C}$ as $d_{min}\left(\mathcal{C}\right) \leq n - k + 1$. Codes which meet the Singleton bound are called maximum distance separable (MDS) codes (see, *e.g.*, [116]).

5.2.1 Linearized Polynomials

In this section, we review some properties of linearized polynomials. (For details, please see [117].) For convenience, we restate the definition of linearized polynomials. Recall that $x^{q^i} = x^{[i]}$.

Definition 10. *[Linearized Polynomial] ([117]) A polynomial in $\mathbb{F}_{q^m}[x]$ of the following form*

$$L(x) = \sum_{i=0}^{n} a_i x^{[i]} \tag{5.1}$$

is called as a linearized polynomial or a q-polynomial over $\mathbb{F}_{q^m}$. Further, $\max\{i \in [n] : a_i \neq 0\}$ *is said to be the q-degree of $L(x)$ denoted as $\textsf{deg}_q\,(L(x))$.*

The name arises from the following property of linearized polynomials, referred to as $\mathbb{F}_q$-linearity ([117]). Let $\mathbb{F}$ be an arbitrary extension field of $\mathbb{F}_{q^m}$ and $L(x)$ be a linearized polynomial over $\mathbb{F}_{q^m}$, then

$$L(\alpha + \beta) = L(\alpha) + L(\beta) \quad \forall\ \alpha, \beta \in \mathbb{F}. \tag{5.2}$$

$$L(c\alpha) = cL(\alpha) \quad \forall\ c \in \mathbb{F}_q \text{ and } \forall\ \alpha \in \mathbb{F}. \tag{5.3}$$

Definition 11. *([117]) [q-Associates] The polynomials*

$$l(x) = \sum_{i=0}^{n} c_i x^i \quad \textit{and} \quad L(x) = \sum_{i=0}^{n} c_i x^{[i]} \tag{5.4}$$

over $\mathbb{F}_{q^m}$ are called q-associates of each other. In particular, $l(x)$ is referred to as the conventional q-associate of $L(x)$ and $L(x)$ is referred to as the linearized q-associate of $l(x)$.

Theorem 12. *[117, Theorem 3.50] Let $L(x)$ be a non-zero linearized polynomial over*

$\mathbb{F}_{q^m}$ and let $\mathbb{F}_{q^s}$ be the extension field of $\mathbb{F}_{q^m}$ that contains all the roots of $L(x)$. Then, the roots form a linear subspace of $\mathbb{F}_{q^s}$, where $\mathbb{F}_{q^s}$ regarded as the vector space over $\mathbb{F}_q$.

The above theorem yields following corollary.

Corollary 1. *Let $L(x)$ be a non-zero linearized polynomial over $\mathbb{F}_{q^m}$ with $\mathsf{deg}_q\left(L(x)\right) = l$, and let $\mathbb{F}_{q^t}$ be arbitrary extension field of $\mathbb{F}_{q^m}$. Then, $L(x)$ has at most l roots in $\mathbb{F}_{q^t}$ that are linearly independet over $\mathbb{F}_q$.*

5.2.2 Rank-Metric Codes

Let $\mathbb{F}_q^{m\times n}$ be the set of all $m \times n$ matrices over $\mathbb{F}_q$. The *rank distance* is a distance measure between elements A and B of $\mathbb{F}_q^{m\times n}$, defined as $d_R\left(A, B\right) = \mathsf{Rank}\left(A - B\right)$. It can be shown that the rank distance is indeed a metric [44]. A rank-metric code is a non-empty subset of $\mathbb{F}_q^{m\times n}$ equipped with the rank distance metric (see [80, 44, 81]). Rank-metric codes can be considered as array codes or matrix codes.

The minimum rank distance of a code $\mathcal{C}$ is given as

$$d_R\left(\mathcal{C}\right) = \min_{C_i,\, C_j \in \mathcal{C},\; C_i \neq C_j} d_R\left(C_i, C_j\right).$$

We denote a linear code $\mathcal{C} \subset \mathbb{F}_q^{m\times n}$ with cardinality $|\mathcal{C}| = (q^m)^k$ and minimum rank distance d as an $(m \times n, k, d)$ code.

The Singleton bound for the rank metric (see [44]) states that every rank-metric code with minimum rank distance d must satisfy

$$|\mathcal{C}| \leq q^{\max\{n,m\}(\min\{n,m\}-d+1)}.$$

Codes that achieve this bound are called maximum rank distance (MRD) codes.

A minimum distance decoder for a rank-metric code $\mathcal{C} \subseteq \mathbb{F}_q^{m\times n}$ takes an array $Y \in$

$\mathbb{F}_q^{m\times n}$ and returns a codeword $X \in \mathcal{C}$ that is closest to Y in rank distance. In other words,

$$X = \arg\min_{X' \in \mathcal{C}} \mathsf{Rank}\left(Y - X'\right). \tag{5.5}$$

Typically, rank-metric codes are considered by leveraging the correspondence between $\mathbb{F}_q^{m\times 1}$ and the extension field $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$. In particular, by fixing a basis for $\mathbb{F}_{q^m}$ as an m-dimensional vector space over $\mathbb{F}_q$, any element of $\mathbb{F}_{q^m}$ can be represented as a length-m vector over $\mathbb{F}_q$. Similarly, any length-n vector over $\mathbb{F}_{q^m}$ can be represented as an $m \times n$ matrix over $\mathbb{F}_q$. The rank of a vector $\mathbf{a} \in \mathbb{F}_{q^m}^n$ is the rank of the corresponding $m \times n$ matrix A over $\mathbb{F}_q$. This rank does not depend on the choice of basis for $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. This correspondence allows us to view a rank-metric code in $\mathbb{F}_q^{m\times n}$ as a block code of length n over $\mathbb{F}_{q^m}$. More specifically, a rank metric code $\mathcal{C} \subseteq \mathbb{F}_q^{m\times n}$ is a block code of length n over $\mathbb{F}_{q^m}$.

Note that, for $m \geq n$, the Singleton bound for rank metric coincides with the classical Singleton bound for the Hamming metric. Indeed, when $m \geq n$, every $(m\times n, k, d)$ MRD code over $\mathbb{F}_q$ is also an $[n, k, d]$ MDS code over $\mathbb{F}_{q^m}$, and hence can correct any $d - 1$ *column erasures*.

Gabidulin Codes

Gabidulin [44] presented a construction of a class of MRD codes for $m \geq n$. The construction is based on the evaluation of a special type of polynomials called *linearized polynomials*. For notational convenience, we write $x^{q^i} = x^{[i]}$.

Definition 12. *[Linearized Polynomial] ([117]) A polynomial in* $\mathbb{F}_{q^m}[x]$ *of the following form*

$$L(x) = \sum_{i=0}^{n} a_i x^{[i]} \tag{5.6}$$

is called a linearized polynomial, or a q-polynomial, over $\mathbb{F}_{q^m}$. *Further,* $\max\{i \in [n] \mid$

$a_i \neq 0\}$ *is said to be the* q*-degree of* $L(x)$*, denoted as* $\mathsf{deg}_q\,(L(x))$.

Gabidulin Code Construction: An (n, k) Gabidulin code over the extension field $\mathbb{F}_{q^m}$ for $m \geq n$ is the set of evaluations of all q-polynomials of q-degree at most $k-1$ over n elements of $\mathbb{F}_{q^m}$ that are linearly independent over $\mathbb{F}_q$.

In particular, let $P = \{p_1, \cdots, p_n\}$ be a set of n elements in $\mathbb{F}_{q^m}$ that are linearly independent over $\mathbb{F}_q$ ($m \geq n$). Let $G_{\mathbf{m}}(x) \in \mathbb{F}_{q^m}[x]$ denote the linearized polynomial of q-degree at most $k-1$ with coefficients $\mathbf{m} = [m_0 \;\, m_1 \;\, \cdots \;\, m_{k-1}] \in \mathbb{F}_{q^m}^k$ as follows.

$$G_{\mathbf{m}}(x) = \sum_{j=0}^{k-1} m_j x^{[j]}, \tag{5.7}$$

Then, the Gabidulin code is obtained by the following evaluation map

$$\begin{aligned} Enc : \mathbb{F}_{q^m}^k &\to \mathbb{F}_{q^m}^n \\ \mathbf{m} &\mapsto \{G_{\mathbf{m}}(\gamma), \gamma \in P\} \end{aligned} \tag{5.8}$$

Therefore, we have

$$\mathcal{C}_{Gab} = \left\{(G_{\mathbf{m}}(\gamma), \gamma \in P) \mid \mathbf{m} \in \mathbb{F}_{q^m}^k\right\}. \tag{5.9}$$

Reed-Solomon Code Construction: It is worth mentioning the analogy between Reed-Solomon codes and Gabidulin codes. An (n, k) Reed-Solomon code over the finite field $\mathbb{F}_q$ for $q \geq n$ is the set of evaluations of all polynomials of degree at most $k-1$ over n distinct elements of $\mathbb{F}_q$. More specifically, let $P = \{p_1, \cdots, p_n\}$ be a set of n distinct elements of $\mathbb{F}_q$ ($q \geq n$). Consider polynomials $g_{\mathbf{m}}(x) \in \mathbb{F}_q[x]$ of the following form

$$g_{\mathbf{m}}(x) = \sum_{j=0}^{k-1} m_j x^j, \tag{5.10}$$

Then, the Reed-Solomon code is obtained by the following evaluation map

$$\begin{aligned} Enc : \mathbb{F}_q^k &\to \mathbb{F}_q^n \\ \mathbf{m} &\mapsto \{g_{\mathbf{m}}(\gamma), \gamma \in P\} \end{aligned} \tag{5.11}$$

Therefore, we have

$$\mathcal{C}_{RS} = \left\{ (g_{\mathbf{m}}(\gamma), \gamma \in P) \mid \mathbf{m} \in \mathbb{F}_q^k \right\}. \tag{5.12}$$

Remark 20. *For the same information vector* $\mathbf{m} = [m_0 \cdots m_{k-1}]$*, the evaluation polynomials of the Gabidulin code and the Reed-Solomon code are* q*-associates of each other. (See Definition 11.)*

5.2.3 Subspace Codes

The set of all subspaces of $\mathbb{F}_q^M$, called the *projective space* of order n over $\mathbb{F}_q$, is denoted by $\mathcal{P}_q(M)$. The set of all n-dimensional subspaces of $\mathbb{F}_q^M$, called a *Grassmannian*, is denoted by $\mathcal{G}_q(M, n)$, where $0 \leq n \leq M$. Note that $\mathcal{P}_q(M) = \cup_{n=0}^{M} \mathcal{G}_q(M, n)$.

In [101], the notion of *subspace distance* was introduced. Let $U, V \in \mathcal{P}_q(M)$. The subspace distance between U and V is defined as

$$d_S(U, V) = \mathsf{dim}(U) + \mathsf{dim}(V) - 2\,\mathsf{dim}(U \cap V). \tag{5.13}$$

It is shown in [101] that the subspace distance is indeed a metric on $\mathcal{P}_q(M)$.

A *subspace code* is a non-empty subset of $\mathcal{P}_q(M)$ equipped with the subspace distance metric [101]. The minimum subspace distance of a subspace code $\Omega \subseteq \mathcal{P}_q(M)$ is defined as

$$d_S(\Omega) = \min_{V_i, V_j \in \Omega,\, V_i \neq V_j} d_S(V_i, V_j). \tag{5.14}$$

A subspace code Ω in which each codeword has the same dimension, say n, *i.e.*, $\Omega \subseteq$

$\mathcal{G}_q(M, n)$, is called a *constant-dimension* code. Such a code with minimum subspace distance d is denoted as an $(M, n, \log_q |\Omega|, d)$ code. It is easy to see, from (5.13) and (5.14), that the minimum distance of a constant-dimension code is always an even number. In the rest of the paper, we restrict our attention to constant-dimension codes.

Construction Based on Lifting Rank-Metric Codes

In [103], the authors presented a construction for a broad class of subspace codes based on rank-metric codes. The construction takes codewords of a rank-metric code and generates codewords of a subspace code using an operation called *lifting*, described in the following.

Definition 13 (Lifting). *Consider the following mapping*

$$\begin{aligned} \Lambda : \mathbb{F}_q^{m\times n} &\to \mathcal{G}_q(m+n, n), \\ X \quad &\mapsto \Lambda(X) = \left\langle \begin{bmatrix} I \\ X \end{bmatrix} \right\rangle, \end{aligned} \tag{5.15}$$

where I is the $n \times n$ identity matrix. The subspace $\Lambda(X)$ is called the lifting of the matrix X. Similarly, for a rank-metric code $\mathcal{C} \subseteq \mathbb{F}_q^{m\times n}$, the subspace code $\Lambda(\mathcal{C}) = \{\Lambda(X) : X \in \mathcal{C}\}$ is called the lifting of $\mathcal{C}$.

Note that the lifting operation $X \mapsto \Lambda(X)$ is an injective mapping since every subspace corresponds to a unique matrix in reduced column echelon form (RCEF). Thus, we have $|\Lambda(\mathcal{C})| = |\mathcal{C}|$. Also, a subspace code constructed by lifting is a constant-dimension code, with each codeword having dimension n.

The key feature of the lifting based construction is that the subspace code constructed by lifting inherits the distance properties of its underlying rank-metric code. More specifically, we have the following result from [103].

Lemma 5. *([103]) Consider a rank-metric code $\mathcal{C} \subseteq \mathbb{F}_q^{m \times n}$,. Then, we have*

$$d_S\left(\Lambda(\mathcal{C})\right) = 2\, d_R\left(\mathcal{C}\right).$$

5.2.4 Codes with Locality

Locality of of a code captures the number of symbols participating in recovering a lost symbol. In particular, an (n, k) code is said to have locality r if every symbol is recoverable from a set of at most r symbols. For linear codes with locality, essentialy a *local* parity check code of length at most $r + 1$ is associated with every symbol. The notion of locality can be generalized to accommodate local codes of larger distance as follows (see [59]).

Definition 14. *[Locality] An (n, k) code $\mathcal{C}$ is said to have (r, δ) locality, if for each symbol $\mathbf{c}_i$, $i \in [n]$, of a codeword $\mathbf{c} = [\mathbf{c}_1\, \mathbf{c}_2\, \cdots \mathbf{c}_n] \in \mathcal{C}$, there exists a set of indices $\Gamma\left(i\right)$ such that*

1. *$i \in \Gamma\left(i\right)$,*

2. *$|\Gamma\left(i\right)| \leq r + \delta - 1$, and*

3. *$d_{min}\left(\mathcal{C}\,|_{\Gamma(i)}\right) \geq \delta$.*

The code $\mathcal{C}\,|_{\Gamma(i)}$ is said to be the local code associated with the i-th coordinate of $\mathcal{C}$.

Properties 2 and 3 imply that for any codeword in $\mathcal{C}$, the values in $\Gamma\left(i\right)$ are uniquely determined by any r of those values. Under Hamming metric, the (r, δ) locality allows one to repair any $\delta - 1$ erasures in $\mathcal{C}\,|_{\Gamma(i)}$, $\forall i \in [n]$, *locally* by accessing at most r other symbols. When $\delta = 2$, the above definition reduces to the classical definition of locality proposed by Gopalan *et al.* [9], wherein any one erasure can be repaired by accessing at most r symbols. The Singleton bound can be generalized to accommodate locality constraints.

In particular, the minimum Hamming distance of an (n, k) code $\mathcal{C}$ with (r, δ) locality is upper bounded as follows (see [22, Theorem 21], also [59, Theorem 2] for linear codes):

$$d_{min}(\mathcal{C}) \leq n - k + 1 - \left(\left\lceil \frac{k}{r} \right\rceil - 1 \right) (\delta - 1). \tag{5.16}$$

Most of the existing work on locally recoverable codes has been focused on locality with respect to the Hamming metric. We are interested in locality with respect to the rank metric.

5.3 Codes with Rank-Locality

Recall from Definition 14 that, for a code $\mathcal{C}$ with (r, δ) locality, the *local code* $\mathcal{C}\mid_{\Gamma(i)}$ associated with the i-th symbol, $i \in [n]$, has minimum distance at least δ. We are interested in rank-metric codes such that the local code associated with every column should be a rank-metric code with minimum rank distance guarantee. This motivates us to generalize the concept of locality to that of *rank-locality* as follows.

Definition 15 (Rank-Locality)**.** *An $(m \times n, k)$ rank-metric code $\mathcal{C}$ is said to have (r, δ) rank-locality if for each column $i \in [n]$ of the codeword matrix, there exists a set of columns $\Gamma(i) \subset [n]$ such that*

1. *$i \in \Gamma(i)$,*

2. *$|\Gamma(i)| \leq r + \delta - 1$, and*

3. *$d_R\left(\mathcal{C}\mid_{\Gamma(i)}\right) \geq \delta$,*

where $\mathcal{C}\mid_{\Gamma(i)}$ is the restriction of $\mathcal{C}$ on the columns of $\Gamma(i)$. The code $\mathcal{C}\mid_{\Gamma(i)}$ is said to be the local code associated with the i-th column. An $(m \times n, k)$ rank-metric code with minimum distance d and (r, δ) locality is denoted as an $(m \times n, k, d, r, \delta)$ rank-metric code.

As we will see in Section 5.5, the (r,δ)-rank-locality allows us to repair any crisscross erasure pattern of weight $\delta-1$ in $\mathcal{C}\mid_{\Gamma(i)}$, $\forall i\in[n]$, *locally* by accessing the symbols of $\mathcal{C}\mid_{\Gamma(i)}$. Further, we can correct any crisscross erasure pattern of weight $d_R(\mathcal{C})-1$ in $\mathcal{C}$ by accessing unerased symbols of $\mathcal{C}$.

Remark 21. *In the remainder of the paper, we assume that the columns of an $(m\times n,k,r,\delta)$ rank-metric code $\mathcal{C}$ can be partitioned into $\mu := n/(r+\delta-1)$ disjoint sets $C_1,\ldots,C_\mu$ each of size $r+\delta-1$ such that, for all $i\in C_j$, $\Gamma(i)=C_j$. In other words, we assume that the local codes associated with the columns have disjoint coordinates.*

5.3.1 Upper Bound on Rank Distance

It is easy to find the Singleton-like upper bound on the minimum rank distance for codes with rank-locality using the results in the Hamming metric.

Theorem 13. *For a rank-metric code $\mathcal{C}\subseteq\mathbb{F}_q^{m\times n}$ of cardinality q^{mk} with (r,δ) rank-locality, we have*

$$d_R(\mathcal{C})\le n-k+1-\left(\left\lceil\frac{k}{r}\right\rceil-1\right)(\delta-1). \tag{5.17}$$

Proof. Note that by fixing a basis for $\mathbb{F}_{q^m}$ as a vector space over $\mathbb{F}_q$, we can obtain a bijection $\phi:\mathbb{F}_{q^m}\to\mathbb{F}_q^{m\times 1}$. This can be extended to a bijection $\phi:\mathbb{F}_{q^m}^n\to\mathbb{F}_q^{m\times n}$. Then, for any vector $\mathbf{c}\in\mathbb{F}_{q^m}^n$, there is a corresponding matrix $C\in\mathbb{F}_q^{m\times n}$ such that $C=\phi(\mathbf{c})$. For any such vector-matrix pair, we have

$$\mathsf{Rank}\,(C)\le\mathsf{wt}\,(\mathbf{c})\,. \tag{5.18}$$

An $(m\times n,k,d)$ rank-metric code $\mathcal{C}$ over $\mathbb{F}_q$ can be considered as a block code of length n over $\mathbb{F}_{q^m}$, denoted as $\mathcal{C}'$. From (5.18), it follows that $d_R(\mathcal{C})\le d_{min}(\mathcal{C}')$. Moreover, it follows that, if $\mathcal{C}$ has (r,δ) rank-locality, then the corresponding code $\mathcal{C}'$ possesses (r,δ) locality in the Hamming metric. Therefore, an upper bound on the minimum Hamming distance of an (n, k, d')-LRC $\mathcal{C}'$ with (r, δ) locality is also an upper bound on the rank distance of an $(m \times n, k, d)$ rank-metric code with (r, δ) rank-locality. Hence, (5.17) follows from (5.16). □

5.3.2 Code Construction

We build upon the construction methodology of Tamo and Barg [25] to construct codes with rank-locality that are optimal with respect to the rank distance bound in (5.17). In particular, the codes are constructed as the evaluations of specially designed linearized polynomials on a specifically chosen set of points of $\mathbb{F}_{q^m}$. The detailed construction is as follows.

Construction 7. *[$(m \times n, k, r, \delta)$ rank-metric code.] Let n, k, r, δ be positive integers such that $r \mid k$, $(r + \delta - 1) \mid n$, and $n \mid m$. Define $\mu := n/(r + \delta - 1)$. Fix $q \geq 2$ to be a power of a prime. Let $\mathcal{A} = \{\alpha 1, \ldots, \alpha r + \delta - 1\}$ be a basis of $\mathbb{F}_{q^{r+\delta-1}}$ as a vector space over $\mathbb{F}_q$, and $\mathcal{B} = \{\beta 1, \ldots, \beta \mu\}$ be a basis of $\mathbb{F}_{q^n}$ as a vector space over $\mathbb{F}_{q^{r+\delta-1}}$. Define the set of n evaluation points $P \subset \mathbb{F}_{q^m}$, with the partition $\mathcal{P} = P_1 \cup \cdots \cup P_\mu$, where $P_j = \{\alpha i \beta j, 1 \leq i \leq r + \delta - 1\}$ for $1 \leq j \leq \mu$. To encode the message $\mathbf{m} \in \mathbb{F}_{q^m}^k$, denoted as $\mathbf{m} = \left\{m_{ij} : i = 0, \ldots, r - 1; j = 0, \ldots, \frac{k}{r} - 1\right\}$, define the encoding polynomial*

$$G_{\mathbf{m}}(x) = \sum_{i=0}^{r-1} \sum_{j=0}^{\frac{k}{r}-1} m_{ij} x^{[(r+\delta-1)j+i]}. \tag{5.19}$$

The codeword for $\mathbf{m}$ is obtained as the vector of the evaluations of $G_{\mathbf{m}}(x)$ at all the points of P. In other words, the linear code $\mathcal{C}_{\text{Loc}}$ is constructed as the following evaluation map:

$$\begin{aligned} Enc : \mathbb{F}_{q^m}^k &\to \mathbb{F}_{q^m}^n \\ \mathbf{m} &\mapsto \{G_{\mathbf{m}}(\gamma), \gamma \in P\}. \end{aligned} \tag{5.20}$$

Therefore, we have

$$\mathcal{C}_{Loc} = \left\{ (G_{\mathbf{m}}(\gamma), \gamma \in P) \mid \mathbf{m} \in \mathbb{F}_{q^m}^k \right\}. \tag{5.21}$$

The $(m \times n, k)$ rank-metric code is obtained by considering the matrix representation of every codeword obtained as above by fixing a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. We denote the following μ codes as the local codes.

$$\mathcal{C}_j = \left\{ (G_{\mathbf{m}}(\gamma), \gamma \in P_j) \mid \mathbf{m} \in \mathbb{F}_{q^m}^k \right\}, \quad 1 \leq j \leq \mu. \tag{5.22}$$

Remark 22. *[Field Size] It is worth mentioning that, even for constructing Gabidulin codes of length n over $\mathbb{F}_{q^m}$, it is required that $m \geq n$ [44]. Note that, it is sufficient to choose $m = n$ and $q = 2$ in our construction. In other words, the field size of 2^n is sufficient for the proposed code construction.*

In the following, we show that Construction 7 gives codes with rank-locality, which are optimal with respect to the rank distance bound in Theorem 13.

Theorem 14. *Construction 7 gives a linear $(m \times n, k, d)$ rank-metric code $\mathcal{C}_{Loc}$ with (r, δ) rank-locality such that the minimum rank distance d is equal to the upper bound given in* (5.17).

Proof. The proof makes use of two key lemmas; Lemma 6 is used to prove the the rank distance optimality and Lemma 7 is used to prove the rank-locality for the proposed construction.

We begin with showing the rank distance optimality of $\mathcal{C}_{Loc}$. The first step is to prove the linear independence of the evaluation points P as stated in the following lemma.

Lemma 6. *The n evaluation points given in Construction 7,*

$$P = \{\alpha i \beta j, 1 \leq i \leq r + \delta - 1, 1 \leq j \leq \mu\},$$

are linearly independent over $\mathbb{F}_q$.

Proof. Suppose, for contradiction, that the evaluation points are linearly dependent over $\mathbb{F}_q$. Then, we have $\sum_{j=1}^{\mu}\sum_{i=1}^{r+\delta-1}\omega_{ij}\alpha i\beta j = 0$ with coefficients $\omega_{ij} \in \mathbb{F}_q$ such that not all ω_{ij}'s are zero. We can write the linear dependence condition as $\sum_{j=1}^{\mu}\left(\sum_{i=1}^{r+\delta-1}\omega_{ij}\alpha i\right)\beta j = 0$. Now, from the linear independence of the βj's over $\mathbb{F}_{q^{r+\delta-1}}$, we have $\sum_{i=1}^{r+\delta-1}\omega_{ij}\alpha i = 0$ for each $1 \leq j \leq \mu$. However, as the αi's are linearly independent over $\mathbb{F}_q$, we have every $\omega_{ij} = 0$. This is a contradiction. □

Lemma 6 essentially asserts that $\mathcal{C}_{Loc}$ is obtained as the evaluations of $G_{\mathbf{m}}(x)$ on n points of $\mathbb{F}_{q^m}$ that are linearly independent over $\mathbb{F}_q$. Combining this with the structure of $G_{\mathbf{m}}(x)$ (see (5.19)), $\mathcal{C}_{Loc}$ can be considered as a subcode of an $\left(n, k + \left(\frac{k}{r} - 1\right)(\delta - 1)\right)$ Gabidulin code (cf. (5.7)). Hence, $d_R\left(\mathcal{C}_{Loc}\right) \geq n - k + 1 - \left(\frac{k}{r} - 1\right)(\delta - 1)$, which shows $d_R\left(\mathcal{C}_{Loc}\right)$ attains the upper bound (5.17) in Theorem 13, and thus that the proposed construction is optimal with respect to rank distance. We present an alternative proof from first principles using the properties of linearized polynomials after this proof for the sake of completeness.

Second, we show that $\mathcal{C}_{Loc}$ has (r, δ) rank-locality. Define $H(x) = x^{q^{r+\delta-1}-1} = x^{[r+\delta-1]-1}$. We note that (5.19) can be written in the following form using $H(x)$.

$$G_{\mathbf{m}}(x) = \sum_{i=0}^{r-1} G_i(x)x^{[i]}, \tag{5.23}$$

where

$$G_i(x) = m_{i0} + \sum_{j=1}^{\frac{k}{r}-1} m_{ij}[H(x)]^{\sum_{l=0}^{j-1} q^{(r+\delta-1)l+i}}. \tag{5.24}$$

To see this, observe that

$$\begin{aligned}
[H(x)]^{\sum_{l=0}^{j-1} q^{(r+\delta-1)l+i}} &= \left[x^{q^{r+\delta-1}-1}\right]^{\sum_{l=0}^{j-1} q^{(r+\delta-1)l+i}} \\
&= x^{\sum_{l=0}^{j-1} q^{(r+\delta-1)(l+1)+i} - \sum_{l=0}^{j-1} q^{(r+\delta-1)l+i}} \\
&= x^{q^{(r+\delta-1)j+i}-q^{i}}. \qquad (5.25)
\end{aligned}$$

Using this in (5.24), we get

$$G_i(x) = m_{i0} + \sum_{j=1}^{\frac{k}{r}-1} m_{ij} x^{[(r+\delta-1)j+i]-[i]}. \qquad (5.26)$$

Substituting (5.26) into (5.23) gives us (5.19).

Now, to prove the rank-locality, we want to show that $d_R\left(\mathcal{C}_j\right) \geq \delta$ for every local code $1 \leq j \leq \mu$. Towards this, let $\gamma \in P_j$ and define the *repair polynomial* as

$$R_j(x) = \sum_{i=0}^{r-1} G_i(\gamma) x^{[i]}. \qquad (5.27)$$

We show that $\mathcal{C}_j$ can be considered as obtained by evaluating $R_j(x)$ on the points of P_j. For this, we first prove that $H(x)$ is constant on all points of P_j for each $1 \leq j \leq \mu$.

Lemma 7. *Consider the partition of the set of evaluation points given in Construction 7 as $\mathcal{P} = P_1 \cup \cdots \cup P_\mu$, where $P_j = \{\alpha i \beta j, 1 \leq i \leq r+\delta-1\}$. Then, $H(x)$ is constant on all evaluation points of any set P_j, $1 \leq j \leq \mu$.*

Proof. Note that $H(\beta j \alpha i) = (\beta j \alpha i)^{[r+\delta-1]-1} = \beta j^{[r+\delta-1]-1} \alpha i^{[r+\delta-1]-1} = \beta j^{[r+\delta-1]-1}$, where the last equality follows from $\alpha i \in \mathbb{F}_{q^{r+\delta-1}} \setminus \{0\}$. Thus, $H(\omega) = \beta j^{[r+\delta-1]-1}$, for all $\omega \in P_j$, $1 \leq j \leq \mu$. □

Note that, since $G_i(x)$ is a linear combination of powers of $H(x)$, it is also constant on the set P_j. In other words, we have

$$G_i(\gamma) = G_i(\lambda), \quad \forall\, \gamma, \lambda \in P_j, \tag{5.28}$$

for every $0 \leq i \leq r-1$.

Moreover, when evaluating $R_j(x)$ in $\lambda \in P_j$, we get

$$R_j(\lambda) = \sum_{i=0}^{r-1} G_i(\gamma)\lambda^{[i]} = \sum_{i=0}^{r-1} G_i(\lambda)\lambda^{[i]} = G_{\mathbf{m}}(\lambda). \tag{5.29}$$

Hence, the evaluations of the encoding polynomial $G_{\mathbf{m}}(x)$ and the repair polynomial $R_j(x)$ on points in P_j are identical. In other words, we can consider that $\mathcal{C}_j$ is obtained by evaluating $R_j(x)$ on points of P_j. Now, since points of P_j are linearly independent over $\mathbb{F}_q$, and $R_j(x)$ is a linearized polynomial of q-degree $r-1$, $\mathcal{C}_j$ can be considered as a $(r+\delta-1, r)$ Gabidulin code (cf. (5.7)). Thus, $\mathcal{C}_j$ is an MRD code, and we have $d_R(\mathcal{C}_j) = \delta$, which proves the rank-locality of the proposed construction.

This concludes the proof of Theorem 14. □

Proof from the First Principles

We note that the result $d_R(\mathcal{C}) \geq \delta$ also follows from Lemma 8 as shown in the following, which is proved from first principles using the properties of linearized polynomials.

We begin with a useful lemma regarding the minimum rank-distance of the rank-metric codes that are obtained through evaluation of linearized polynomials.

Lemma 8. *Let P be a set of n elements in $\mathbb{F}_{q^m}$ that are linearly independent over $\mathbb{F}_q$ ($m \geq n$). Consider a linearized polynomial $L_{\mathbf{m}}(x) \in \mathbb{F}_{q^m}[x]$ of the following form*

$$L_{\mathbf{m}}(x) = \sum_{j=1}^{k} m_{i_j} x^{[i_j]}, \tag{5.30}$$

where i_j's are non-negative integers such that $0 \leq i_1 < i_2 < \cdots < i_k \leq n-1$, and $k \leq n$. Consider the code obtained by the following evaluation map

$$\begin{aligned} Enc : \mathbb{F}_{q^m}^k &\to \mathbb{F}_{q^m}^n \\ \mathbf{m} &\mapsto \{L_{\mathbf{m}}(\gamma), \gamma \in P\} \end{aligned} \tag{5.31}$$

In other words, we have

$$\mathcal{C} = \left\{ L_{\mathbf{m}}(\gamma) \mid \mathbf{m} \in \mathbb{F}_{q^m}^k, \gamma \in P \right\}. \tag{5.32}$$

Then, $\mathcal{C}$ is a linear $[m \times n, k, d]$ rank-metric code with rank-distance $d \geq n - i_k$.

Proof. First, note that a codeword $\mathbf{c} \in \mathcal{C}$ is the evaluation of $L_{\mathbf{m}}(x)$ on n points of P for a fixed $\mathbf{m} \in \mathbb{F}_{q^m}^k$. Thus, a codeword is a set of n values each in $\mathbb{F}_{q^m}$. By fixing a basis for $\mathbb{F}_{q^m}$ as a vector space over $\mathbb{F}_q$, we can represent a codeword $\mathbf{c} \in \mathbb{F}_{q^m}^n$ as an $m \times n$ matrix $C \in \mathbb{F}_q^{m \times n}$. Thus, $\mathcal{C}$ is a matrix or array code.

Second, note that $\mathcal{C}$ is an evaluation map over $\mathbb{F}_{q^m}$. Observe that $\mathbf{m} \mapsto L_{\mathbf{m}}(x)$ is an injective map. Since q-degree of $L_{\mathbf{m}}(x)$ is at most $n-1$, two distinct polynomials $L_{\mathbf{m}_j}(x)$ and $L_{\mathbf{m}_l}(x)$ result in distinct codewords, and thus, dimension of the code (over $\mathbb{F}_{q^m}$) is k.

Finally, we show that $d_R(\mathcal{C}) \geq n - i_k$. Notice that

$$\max_{L_{\mathbf{m}}, \mathbf{m} \in \mathbb{F}_{q^m}^k} \mathsf{deg}_q (L_{\mathbf{m}}) \leq i_k, \tag{5.33}$$

where $\mathsf{deg}_q(F)$ denotes the q-degree of a linearized polynomial F.

Consider a codeword $\mathbf{c}$ as a length-n vector over $\mathbb{F}_{q^m}$. Let $\mathbf{m_c}$ be the message vector resulting in $\mathbf{c}$, and $L_{\mathbf{m_c}}$ be the corresponding polynomial giving $\mathbf{c}$. Let $C \in \mathbb{F}_q^{m \times n}$ be the matrix representation of $\mathbf{c}$ for some basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Suppose $\mathsf{Rank}(C) = w_r$. We want to prove that $w_r \geq n - i_k$. Suppose, for contradiction, $w_r < n - i_k$.

Let $\mathsf{wt}(\mathbf{c}) = w$. Clearly, $w_r \leq w$. Without loss of generality (WLOG), assume that the last $n - w$ columns of C are zero. We know that $n - w$ points in P, $\{\gamma_{w+1}, \ldots, \gamma_n\}$, are the roots of $L_{\mathbf{m_c}}(x)$. Note that, since elements of P are linearly independent over $\mathbb{F}_q$, $w \geq n - i_k$ (see Corollary 1).

WLOG, assume that the first w_r columns of C are linearly independent over $\mathbb{F}_q$. After doing column operations, we can make the middle $w - w_r$ columns as zero columns. Thus, there exist coefficients c_j^l's, not all zero, such that

$$\sum_{j=1}^{w_r} c_j^l L_{\mathbf{m_c}}(\gamma_j) + c_{j+1}^l L_{\mathbf{m_c}}(\gamma_{w_r+l}) = 0, \quad \text{for } 1 \leq l \leq w - w_r. \tag{5.34}$$

By using $\mathbb{F}_q$-linearity property of linearized polynomials (see (5.2), (5.3)), the above set of equations (5.34) is equivalent to

$$L_{\mathbf{m_c}}\left(\sum_{j=1}^{w_r} c_j^l \gamma_j + c_{j+1}^l \gamma_{w_r+l}\right) = 0, \quad \text{for } 1 \leq l \leq w - w_r. \tag{5.35}$$

Therefore, $\left\{\sum_{j=1}^{w_r} c_j^l \gamma_j + c_{j+1}^l \gamma_{w_r+l}, 1 \leq l \leq w - w_r\right\}$ are also the roots of $L_{\mathbf{m_c}}(x)$. Together with $\{\gamma_{w+1}, \ldots, \gamma_n\}$ as its roots, $L_{\mathbf{m_c}}(x)$ has $n - w_r > i_k$ roots. Note that, since γ_j's are linearly independent over $\mathbb{F}_q$, so are all of the $n - w_r$ roots. Thus, $L_{\mathbf{m_c}}(x)$ has more than i_k roots that are linearly independent over $\mathbb{F}_q$, which is a contradiction due to (5.33) and Corollary 1. □

From the above lemma, it follows that $\mathcal{C}$ obtained using Construction 7 is a linear $(m \times n, k)$ rank-metric code. Observe that the q-degree of $G_{\mathbf{m}}(x)$ is bounded as

$$\mathsf{deg}_q\left(G_{\mathbf{m}}(x)\right) \leq \left(\frac{k}{r} - 1\right)(r + \delta - 1) + r - 1 = k - 1 + \left(\frac{k}{r} - 1\right)(\delta - 1).$$

Further, we have $d_R(\mathcal{C}) \geq n-k+1-\left(\frac{k}{r}-1\right)(\delta-1)$, which proves the rank-distance optimality.

Example for the Proposed Construction

Next, we present an example of an $(9\times 9, 4)$ rank-metric code with $(2,2)$ rank-locality. We note that the code presented in this example satisfies the correctability constraints specified in the motivating example (Example 5) in the introduction section.

Example 6. *Let $n=9, k=4, r=2, \delta=2$. Set $q=2$ and $m=n$. Let ω be the primitive element of $\mathbb{F}_{2^9}$ with respect to the primitive polynomial $p(x)=x^9+x^4+1$. Note that ω^{73} generates $\mathbb{F}_{2^3}$, as $(\omega^{73})^7=1$. Consider $\mathcal{A}=\{1,\omega^{73},\omega^{146}\}$ as a basis for $\mathbb{F}_{2^3}$ over $\mathbb{F}_2$. We view $\mathbb{F}_{2^9}$ as an extension field over $\mathbb{F}_{2^3}$ considering the irreducible polynomial $p(x)=x^3+x+\omega^{73}$. It is easy to verify that ω^{309} is a root of $p(x)$, and thus, $\mathcal{B}=\{1,\omega^{309},\omega^{107}\}$ forms a basis of $\mathbb{F}_{2^9}$ over $\mathbb{F}_{2^3}$. Then, the evaluation points P and their partition $\mathcal{P}$ is as follows.*

$$\mathcal{P}=\{P_1=\{1,\omega^{73},\omega^{146}\},\ \ P_2=\{\omega^{309},\omega^{382},\omega^{455}\},\ \ P_3=\{\omega^{107},\omega^{180},\omega^{253}\}\}.$$

Let $\mathbf{m}=(m_{00},\ m_{01},\ m_{10},\ m_{11})\in\mathbb{F}_{2^9}^4$ be the information vector. Define the encoding polynomial (as in (5.19)) as follows.

$$G_{\mathbf{m}}(x)=m_{00}x^{[0]}+m_{01}x^{[3]}+m_{10}x^{[1]}+m_{11}x^{[4]}.$$

The codeword $\mathbf{c}$ for the information vector $\mathbf{m}$ is obtained as the evaluation of the polynomial $G_{\mathbf{m}}(x)$ at all the points of P. The code $\mathcal{C}$ is the set of codewords corresponding to all $\mathbf{m}\in\mathbb{F}_{2^9}^4$.

From Lemma 6, the evaluation points are linearly independent over $\mathbb{F}_2$, and thus, $\mathcal{C}$ can be considered as a subcode of a $(9,5)$ Gabidulin code (cf. (5.7)). Thus, $d_R(\mathcal{C})=5$, *which is optimal with respect to* (5.17).

Now, consider the local codes $\mathcal{C}_j$, $1 \leq j \leq 3$. *It is easy to verify that* $\mathcal{C}_j$ *can be obtained by evaluating the repair polynomial* $R_j(x)$ *on* P_j *given as follows (see* (5.27)*).*

$$R_1(x) = (m_{00} + m_{01})x^{[0]} + (m_{10} + m_{11})x^{[1]}, \tag{5.36}$$

$$R_2(x) = (m_{00} + \omega^{119} m_{01})x^{[0]} + (m_{10} + \omega^{238} m_{11})x^{[1]}, \tag{5.37}$$

$$R_3(x) = (m_{00} + \omega^{238} m_{01})x^{[0]} + (m_{10} + \omega^{476} m_{11})x^{[1]}. \tag{5.38}$$

For instance, let the message vector be $\mathbf{m} = (\omega, \omega^2, \omega^4, \omega^8)$. *Then, the codeword is*

$$\mathbf{c} = (\omega^{440}, \omega^{307}, \omega^{81}, \omega^{465}, \omega^{11}, \omega^{174}, \omega^{236}, \omega^{132}, \omega^{399}).$$

One can easily check that evaluating $R_1(x)$ *on* P_1 *gives* $\mathbf{c}_1 = (\omega^{440}, \omega^{307}, \omega^{81})$*, evaluating* $R_2(x)$ *on* P_2 *gives* $\mathbf{c}_2 = (\omega^{465}, \omega^{11}, \omega^{174})$*, and evaluating* $R_3(x)$ *on* P_3 *gives* $\mathbf{c}_3 = (\omega^{236}, \omega^{132}, \omega^{399})$.

This implies that the local code $\mathcal{C}_j$, $1 \leq j \leq 3$, *can be considered as obtained by evaluating a linearized polynomial of the form* $R_j(x) = m'_0 x^{[0]} + m'_1 x^{[1]}$ *on three points that are linearly independent over* $\mathbb{F}_2$. *Hence,* $\mathcal{C}_j$ *is a Gabidulin code of length 3 and dimension 2, which gives* $d_R(\mathcal{C}\mid_{P_i}) = 2$. *This shows that* $\mathcal{C}$ *has* $(2, 2)$ *rank-locality.*

Comparison with Tamo and Barg [25]

The key idea in [25] is to construct codes with locality as evaluations of a specially designed polynomial over a specifically chosen set of elements of the underlying finite field. To point out the similarities and differences, we briefly review Construction 8 from [25]. We assume that $r \mid k$, and $r + \delta - 1 \mid n$.

Construction 8 from [25]: Let $\mathcal{P} = \{P_1, \ldots, P_\mu\}$, $\mu = n/(r + \delta - 1)$, be a partition of the set $P \subset \mathbb{F}_q$, $|P| = n$, such that $|P_i| = r + \delta - 1$, $1 \leq i \leq \mu$. Let $h \in \mathbb{F}_q[x]$ be a polynomial of degree $r+\delta-1$, called the *good polynomial*, that is constant on each of the sets P_i. For an information vector $\mathbf{m} \in \mathbb{F}_q^k$, define the encoding polynomial

$$g_{\mathbf{m}}(x) = \sum_{i=0}^{r-1} \left(\sum_{j=0}^{\frac{k}{r}-1} m_{ij} h(x)^j \right) x^i.$$

The code $\mathcal{C}$ is defined as the set of n-dimensional vectors

$$\mathcal{C} = \left\{ (g_{\mathbf{m}}(\gamma), \gamma \in P) \mid \mathbf{m} \in \mathbb{F}_q^k \right\}.$$

The authors show that $h(x) = x^{r+\delta-1}$ can be used as a *good polynomial*, when the evaluation points are the cosets of a multiplicative subgroup of $\mathbb{F}_q^*$ of order $r+\delta-1$. In this case, we can write $g_{\mathbf{m}}(x)$ as

$$g_{\mathbf{m}}(x) = \sum_{i=0}^{r-1} \sum_{j=0}^{\frac{k}{r}-1} m_{ij} x^{(r+\delta-1)j+i}. \tag{5.39}$$

Therefore, $\mathcal{C}$ can be considered as a subcode of an $\left(n, k + \left(\frac{k}{r}-1\right)(\delta-1)\right)$ Reed-Solomon code. In addition, local codes $\mathcal{C}_j = \left\{ (g_{\mathbf{m}}(\gamma), \gamma \in P_j) \mid \mathbf{m} \in \mathbb{F}_q^k \right\}$, $1 \leq j \leq \mu$, can be considered as $(r+\delta-1, r)$ Reed-Solomon codes.

In our case, the code $\mathcal{C}_{Loc}$ obtained from Construction 7 can be considered as a subcode of a $\left(n, k + \left(\frac{k}{r}-1\right)(\delta-1)\right)$ Gabidulin code. Further, the local codes $\mathcal{C}_j$, $1 \leq j \leq \mu$, can be considered as $(r+\delta-1, r)$ Gabidulin codes. In fact, as one can see from the proof of Theorem 14, we implicitly use $H(x) = x^{[r+\delta-1]-1}$ as the good polynomial, which evaluates as a constant on all points of P_j for $1 \leq j \leq \mu$ given in Construction 7. It is worth mentioning that (5.39) and (5.19) turn out to be q-associates of each other. (See Definition 11 in Sec. 5.2.1.)

Comparison with Silberstein *et al.* [24]

In [24] (see also [22]), the authors have presented a construction of LRC codes based on rank-metric codes. The idea is to precode the information vector with an $(r\mu, k)$ Gabidulin code over $\mathbb{F}_{q^m}$. The symbols of the codeword are partitioned into μ sets $C_1, \ldots, C_\mu$ of size r each. For each set C_j, an $(r + \delta - 1, r)$ Reed-Solomon code over $\mathbb{F}_q$ is used to obtain $\delta - 1$ local parities, which together with the symbols of C_j forms the codeword of a local code $\mathcal{C}_j$. This ensures that each local code has minimum distance δ. However, it does not guarantee that the minimum rank distance of a local code is at least δ.

In fact, for any $\mathbf{c} \in \mathcal{C}_j$, $1 \leq j \leq \mu$, we have $\mathsf{Rank}\left(\mathbf{c}\right) \leq r$, as the local parities are obtained via linear combinations over $\mathbb{F}_q$. Clearly, when $\delta > r$, the construction cannot achieve rank-locality. Moreover, even if $\delta \leq r$, it is possible to have a codeword $\mathbf{c} \in \mathcal{C}_j$ such that $\mathsf{Rank}\left(\mathcal{C}_i\right) < \delta$ for some local code $\mathcal{C}_j$. Therefore, in general, the construction of [24], that uses Gabidulin codes as outer codes, does not guarantee that the codes possess rank-locality.

On the other hand, our construction can be viewed as a method to design (n, k) linear codes over $\mathbb{F}_{q^m}$ with (r, δ) locality (under Hamming metric). For the construction in [24], the field size of q^n is sufficient for $q \geq r + \delta - 1$ when $\delta > 2$, while one can choose any $q \geq 2$ when $\delta = 2$. When our construction is used to obtain LRCs, it is sufficient to operate over the field of size 2^n.

5.4 Codes with Subspace-Locality

In this section, we extend the concept of locality to that of *subspace-locality*. We begin with setting up the necessary notation. Let $\Omega \subseteq \mathcal{G}_q\left(M, n\right)$ be a constant-dimension subspace code. For every codeword $U \in \Omega$, consider an $M \times n$ matrix $[U]$ in a reduced column echelon form (RCEF) such that columns of $[U]$ span U. In other words, $[U] = \mathsf{rcef}\left([U]\right)$ and $U = \langle [U] \rangle$. Note that columns of $[U]$ form an ordered basis of U.

For a set $S \subset [n]$, let $[U] \mid_S$ denote the $M \times |S|$ sub-matrix of $[U]$ consisting of the columns of $[U]$ indexed by S. Let $U \mid_S = \langle [U] \mid_S \rangle$, and $\Omega \mid_S = \{U \mid_S : U \in \Omega\}$. Note that the code $\Omega \mid_S$ is essentially obtained by taking a projection of every subspace U of Ω on the subspace formed by the basis vectors in S.

Now, we define the notion of subspace-locality in the following.

Definition 16 (Subspace-Locality)**.** *A constant-dimension subspace code $\Omega \subseteq \mathcal{G}_q(M, n)$ is said to have (r, δ) subspace-locality if, for each $i \in [n]$, there exists a set $\Gamma(i) \subset [n]$ such that*

1. $i \in \Gamma(i)$*,*

2. $|\Gamma(i)| \leq r + \delta - 1$*,*

3. *dim* $\left(\Omega \mid_{\Gamma(i)}\right) = |\Gamma(i)|$*, and*

4. $d_S\left(\Omega \mid_{\Gamma(i)}\right) \geq \delta$*.*

The code $\Omega \mid_{\Gamma(i)}$ is said to be the local code associated with the i-th basis vector for the subspaces of Ω. A subspace code $\Omega \subseteq \mathcal{G}_q(M, n)$ with minimum distance d and (r, δ) locality is denoted as an $(M, n, \log_q |\Omega|, d, r, \delta)$ subspace code.

Next, we show that the lifting construction given in (5.15) preserves the locality property.

Lemma 9. *A subspace code obtained by lifting a rank-metric code with (r, δ) rank-locality has $(r, 2\delta)$ subspace-locality.*

Proof. Let $\mathcal{C} \subseteq \mathbb{F}_q^{m \times n}$ be a rank-metric code with (r, δ) rank-locality. For each $i \in [n]$, there is a local code $\mathcal{C} \mid_{\Gamma(i)}$ such that $d_R\left(\mathcal{C} \mid_{\Gamma(i)}\right) \geq \delta$ due to the (r, δ) rank-locality of $\mathcal{C}$.

Let $\Omega = \Lambda(\mathcal{C})$ be the subspace code obtained by lifting $\mathcal{C}$. Let $\Omega\mid_{\Gamma(i)} = \{U\mid_{\Gamma(i)} : U \in \Omega\}$. Consider a pair of codewords $V, V' \in \Omega\mid_{\Gamma(i)}$. Then, we have

$$V = \left\langle \begin{bmatrix} \hat{I}_{\Gamma(i)} \\ C_{\Gamma(i)} \end{bmatrix} \right\rangle, \quad V' = \left\langle \begin{bmatrix} \hat{I}_{\Gamma(i)} \\ C'_{\Gamma(i)} \end{bmatrix} \right\rangle,$$

where $\hat{I}_{\Gamma(i)}$ is an $n \times |\Gamma(i)|$ sub-matrix of the $n \times n$ identity matrix composed of the columns indexed by $\Gamma(i)$, and $C_{\Gamma(i)}, C'_{\Gamma(i)} \in \mathcal{C}\mid_{\Gamma(i)}$. Note that $\mathsf{dim}(V) = \mathsf{dim}(V') = |\Gamma(i)|$. Thus, we have

$$\begin{aligned} d_S(V, V') &\overset{(a)}{=} 2\,\mathsf{dim}(V + V') - \mathsf{dim}(V) - \mathsf{dim}(V') \\ &\overset{(b)}{=} 2\,\mathsf{dim}(V + V') - 2|\Gamma(i)| \\ &\overset{(c)}{=} 2\,\mathsf{Rank}\left(\begin{bmatrix} \hat{I}_{\Gamma(i)} & \hat{I}_{\Gamma(i)} \\ C_{\Gamma(i)} & C'_{\Gamma(i)} \end{bmatrix} \right) - 2|\Gamma(i)| \\ &= 2\,\mathsf{Rank}\left(\begin{bmatrix} \hat{I}_{\Gamma(i)} & 0 \\ C_{\Gamma(i)} & C'_{\Gamma(i)} - C_{\Gamma(i)} \end{bmatrix} \right) - 2|\Gamma(i)| \\ &= 2\,\mathsf{Rank}\left(C'_{\Gamma(i)} - C_{\Gamma(i)}\right) \\ &\overset{(d)}{\geq} 2\delta, \end{aligned} \tag{5.40}$$

where (a) follows from (5.13) and the fact that $\mathsf{dim}(V + V') = \mathsf{dim}(V) + \mathsf{dim}(V') - \mathsf{dim}(V \cap V')$, (b) follows due to $\mathsf{dim}(V) = \mathsf{dim}(V') = |\Gamma(i)|$, (c) follows from the fact that for any pair of matrices X and Y, we have

$$\left\langle \begin{bmatrix} X \\ Y \end{bmatrix} \right\rangle = \langle X \rangle + \langle Y \rangle,$$

and (e) follows from $d_R\left(\mathcal{C}\mid_{\Gamma(i)}\right) \geq \delta$.

The result is immediate from (5.40). □

Now, by lifting rank-metric codes obtained via Construction 7, we get a family of subspace codes with locality. Specifically, from Lemmas 5 and 9, we get the following result as a corollary.

Corollary 2. *Let $\mathcal{C}_{Loc}$ be an $(m \times n, k, d, r, \delta)$ rank-metric code obtained by Construction 7. The code $\Lambda(\mathcal{C}_{Loc})$ obtained by lifting $\mathcal{C}_{Loc}$ is an $(m + n, n, mk, 2d, r, 2\delta)$ subspace code.*

5.5 Correction Capability of Codes with Rank-Locality

Suppose the encoded data is stored on an $m \times n$ array C using an $(m \times n, k, d, r, \delta)$ rank-metric code $\mathcal{C}$ over $\mathbb{F}_{q^m}$. Let $\mathcal{C}_1, \ldots, \mathcal{C}_\mu$ be the local codes of $\mathcal{C}$ and let $C_1, \ldots, C_\mu$ be the corresponding local arrays, where a local array C_i is of size $m \times (n/\mu)$. Our goal is to characterize the class of (possibly correlated) mixed erasure and error patterns corresponding to column and row failures of C that $\mathcal{C}$ can correct locally or globally. Towards this, we consider the notion of crisscross weight of an erasure pattern.

Let $E = [e_{i,j}]_{1 \leq i \leq m, 1 \leq j \leq n}$ be an $m \times n$ binary matrix that specifies the location of the erased symbols of C, referred to as an erasure matrix. In particular, $e_{ij} = 1$ if the (i, j)-th entry of C is erased, otherwise $e_{ij} = 0$. For simplicity, we denote the erasure pattern by E itself. We denote by $E(C_j)$ the n/μ columns of E corresponding to the local array C_j, and we refer to $E(C_j)$ as the erasure pattern restricted to the local array C_j. We first consider the notion of a *cover* of E, which is used to define the crisscross weight of E (see [81], also [82]).

Definition 17. *[Cover of E] A cover of an $m \times n$ matrix E is a pair (X, Y) of sets $X \subseteq [m]$, $Y \subseteq [n]$, such that $e_{ij} \neq 0 \implies ((i \in X)$ or $(j \in Y))$ for all $1 \leq i \leq m$, $1 \leq j \leq n$. The size of the cover (X, Y) is defined as $|(X, Y)| = |X| + |Y|$.*

We define the crisscross weight of an erasure pattern as the crisscross weight of the associated binary matrix E defined as follows.

Definition 18. *[Crisscross weight of E] The crisscross weight of an erasure pattern E is the minimum size $|(X, Y)|$ over all possible covers (X, Y) of the associated binary matrix E. We denote the crisscross weight of E as $\mathsf{wt}_c(E)$.*

Note that a minimum-size cover of a given matrix E is not always unique. Further, the minimum size of a cover of a binary matrix is equal to the maximum number of 1's that can be chosen in that matrix such that no two are on the same row or column [118, Theorem 5.1.4].

Let $E' = [e'_{i,j}]_{1\le i\le m, 1\le j\le n} \in \mathbb{F}_q^{m\times n}$ be a matrix that specifies the location and values of errors occurred in the array, referred to as an error matrix. Specifically, $e'_{i,j} \in \mathbb{F}_q$ denotes the error at the i-th row and the j-th column. If there is no error, $e'_{i,j} = 0$. We assume that for every $1 \le i \le m, 1 \le j \le n$, such that $e_{i,j} = 1$, we have $e'_{i,j} = 0$. In other words, the value of the error is zero at a location where erasure occurs. We denote by $E'(C_j)$ the n/μ columns of E' corresponding to the local array C_j, and we refer to $E'(C_j)$ as the error pattern restricted to the local array C_j.

We characterize erasure and error patterns that $\mathcal{C}$ can correct locally or globally. Define a binary variable δ_j for $1 \le j \le \delta$ as follows.

$$\delta_j = \begin{cases} 1 & \text{if } 2\,\mathsf{Rank}\left(E'(C_j)\right) + \mathsf{wt}_c\left(E(C_j)\right) \le \delta - 1, \\ 0 & \text{otherwise.} \end{cases} \tag{5.41}$$

Recall that, for simplicity, we assume that the local codes associated with columns are disjoint in their support. We note that the proposed construction indeed results in disjoint local codes.

Proposition 5. *Let $\mathcal{C}$ be an $(m \times n, k, d)$ rank-metric code with (r, δ) rank-locality. Let $\mathcal{C}_j$, $1 \leq j \leq \mu$, be the j-th local $(r + \delta - 1, r, \delta)$ rank-metric code, and let C_j be the corresponding local array. Consider erasure and error matrices E and E'. The code $\mathcal{C}_j$ is guaranteed to correct the erasures and errors $E(C_j)$ and $E'(C_j)$ locally by accessing the unerased symbols only from C_j provided*

$$2\,\textsf{\textit{Rank}}\,(E'(C_j)) + \textsf{\textit{wt}}_c\,(E(C_j)) \leq \delta - 1. \tag{5.42}$$

Further, the code $\mathcal{C}$ is guaranteed to correct E and E' provided

$$2\,\textsf{\textit{Rank}}\,(E') + \textsf{\textit{wt}}_c\,(E) - \sum_{j=1}^{\mu} \delta_j \left(2\,\textsf{\textit{Rank}}\,(E'(C_j)) + \textsf{\textit{wt}}_c\,(E(C_j))\right) \leq d - 1, \tag{5.43}$$

where δ_j is defined in (5.41).

Proof. The proof essentially follows from the fact that a rank-metric code $\mathcal{C}$ of rank distance d can correct any erasure pattern E and error pattern E' such that $2\,\textsf{Rank}\,(E') + \textsf{wt}_c\,(E) \leq d - 1$. To see this, consider a minimum-size cover (X, Y) of E. Delete the rows and columns indexed respectively by X and Y in all the codeword matrices of $\mathcal{C}$ as well as from E''. The resulting array code composed of matrices of size $m - |X| \times n - |Y|$ has rank distance at least $d - \textsf{wt}_c\,(E)$. This code can correct any error pattern E'' such that $\textsf{Rank}\,(E'') \leq (d - \textsf{wt}_c\,(E) - 1)/2$ using the minimum distance decoder (cf. (5.5)). This immediately gives (5.42). First correcting erasures and errors locally using $\mathcal{C}_j$ for each $1 \leq j \leq \mu$, and then globally using $\mathcal{C}$ yields (5.43). □

Example 7. *Suppose the data is to be stored on a 9×9 bit array C using the $(9 \times 9, 5, 5, 2, 2)$ rank-metric code discussed in Example 6. Note that the first three columns of C form the first local array C_1, the next three columns form the second local array C_2, and the remaining three columns form the third local array C_3. The encoding satisfies*

??	??	??	??	??	??	?	?	?
??	??	??	??	$c_{2,5}$	$c_{2,6}$	$c_{2,7}$	$c_{2,8}$	$c_{2,9}$
??	??	??	??	$c_{3,5}$	$c_{3,6}$	$c_{3,7}$	$c_{3,8}$	$c_{3,9}$
$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	??	$c_{4,5}$	$c_{4,6}$	$c_{4,7}$	$c_{4,8}$	$c_{4,9}$
$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	??	$c_{5,5}$	$c_{5,6}$	$c_{5,7}$	$c_{5,8}$	$c_{5,9}$
$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	??	$c_{6,5}$	$c_{6,6}$	$c_{6,7}$	$c_{6,8}$	$c_{6,9}$
$c_{7,1}$	$c_{7,2}$	$c_{7,3}$	??	$c_{7,5}$	$c_{7,6}$	$c_{7,7}$	$c_{7,8}$	$c_{7,9}$
$c_{8,1}$	$c_{8,2}$	$c_{8,3}$	??	$c_{8,5}$	$c_{8,6}$	$c_{8,7}$	$c_{8,8}$	$c_{8,9}$
$c_{9,1}$	$c_{9,2}$	$c_{9,3}$	??	$c_{9,5}$	$c_{9,6}$	$c_{9,7}$	$c_{9,8}$	$c_{9,9}$

Figure 5.3: An example of a 9×9 bit array.

the correctability constraints mentioned in Example 5. We give a few examples of erasure patterns that are correcrable in Fig. 5.3, where locally correctable erasures are denoted as '?', while globally correctable erasures are denoted as '??'.

Remark 23. *In Proposition 5, we only characterize the erasure patterns that are locally or globally correctable. It is interesting to consider efficient decoding algorithms on the lines of [119, 39].*

Remark 24. *We note that an $(m \times n, k, d, r, \delta)$ code may correct a number of erasure patterns that are not covered by the class mentioned in Proposition 5. This is analogous to the fact that an LRC can correct a large number of erasures beyond minimum distance. In fact, the class of LRCs that have the maximum erasure correction capability are known as maximally recoverable codes. Along similar lines, it is interesting to extend the notion of maximal recoverability for the rank metric and characterize all the erasure patterns that an $(m \times n, k, d, r, \delta)$ rank-metric code can correct.*

5.6 Conclusion

We constructed codes with locality constraints in the rank and subspace metrics (instead of the conventional Hamming metric). We showed that the proposed local rank-metric codes can recover locally from *crisscross erasures and errors*, which affect a limited number of rows and/or columns of the storage system. We proved a tight Singleton-like upper bound on the minimum rank-distance of linear codes with *rank-locality* constraints. Then, we constructed a family of locally recoverable rank-metric codes that achieve this bound for a broad range of parameters. Finally, we constructed a class of subspace codes with locality constraints in the subspace metric. The construction used the lifting method, and the main idea was to show that a subspace code with locality can be easily constructed from a rank-metric code with locality by using the lifting method.

6. LATENCY ANALYSIS FOR CODES WITH AVAILABILITY*

6.1 Introduction

Cloud storage networks utilize large-scale distributed storage systems (DSS) to handle enormous amounts of data. DSS are expected to provide timely service regardless of the system conditions and loads. The key challenges in cloud storage systems are to maintain reliability against failures, and guarantee high content availability. Reliability is achieved by introducing some form of redundancy using either replication or erasure coding. In order to ensure high availability, the objective is to minimize the delay to serve the download requests arriving at the distributed storage system.

In the past few years, there has been a significant research in investigating how the redundancy used for enhancing reliability can be leveraged to reduce download latency. The main focus has been on analyzing download delay in DSS (or, in general, distributed systems) that employ replication or erasure coding. The class of erasure codes that have received the primary attention are maximum distance separable (MDS) codes. It is well known that MDS codes achieve high storage efficiency as compared to replication. Their utility in reducing the download latency has been studied recently in several works, see *e.g.* [122, 7, 123, 124, 125, 126, 127, 128, 129, 73].

On one hand, these results demonstrate that the download latency in DSS can be improved by employing an MDS coding scheme. On the other hand, the challenge of quickly repairing failed nodes has motivated coding theorists to diverge from MDS codes in search of several novel erasure codes that are *repair efficient* [8, 20, 9, 25, 130, 22]. These coding

*Parts of this chapter are reprinted with permission from [120] "Analyzing the download time of availability codes," by S. Kadhe, E. Soljanin, and A. Sprintson, 2015, *In Proceedings of 2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 1467-1471, June 2015 and [121] "When do the availability codes make the stored data more available?" by S. Kadhe, E. Soljanin, and A. Sprintson, 2015. *In Proceedings of 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 956-963, Sept 2015. Copyright © by IEEE.

schemes have not only made a significant impact on the theory side, but also on the practical side by becoming a part of real systems [131, 78]. The goal of this chapter is to analyze how some these novel erasure codes affect the download latency in DSS. In particular, we focus on *locally repairable codes* with *availability* property.

Locally repairable codes (LRCs) [20, 9] are a class of distributed storage codes, in which a systematic storage node (*i.e.*, a node that stores an uncoded data symbol) can be reconstructed by accessing a group of other storage nodes, called a repair group. LRCs possess a key property that, for each systematic node, the size of its repair group is small. *Availability codes*, which belong to a sub-class of LRCs, have an additional property that each systematic node has multiple, disjoint repair groups [22, 130, 30, 132].

For instance, consider the $(n = 7, k = 3)$ Simplex code that encodes three files, say $\{f_1, f_2, f_3\}$, into seven files given as $\{f_1, f_2, f_3, f_1 + f_2, f_1 + f_3, f_2 + f_3, f_1 + f_2 + f_3\}$. We assume that each file consists of one symbol in a finite field, and the encoded files are stored on seven independent storage nodes. This code is said to have *availability* three as each file has three disjoint repair groups. For example, file f_1 can be can be recovered by reading both f_2 and $f_1 + f_2$ from nodes 2 and 4 or by reading both f_3 and $f_1 + f_3$ from nodes 3 and 5, or by reading $f_2 + f_3$ and $f_1 + f_2 + f_3$ from nodes 6 and 7. Further, the *locality* of each read is at most two – one needs to read at most two files to reconstruct the desired file.

In this chapter, we analyze the delay for downloading individual files from a DSS that employs availability codes. One of our goals is to compare the availability codes with alternative approaches, in particular, the replication schemes and MDS codes. We also strive to understand the role that various parameters of availability coded systems play in the file download time. Moreover, we focus on the problem of *service capacity provisioning* across nodes with the objective of reducing the file download delay. If one can increase the number of parallel reads, say by increasing the number of disjoint repair groups, then the download latency can be reduced. However, increasing the number of repair groups involves increasing the total service capacity of the system, and in practice, available total service capacity is often limited. Therefore, we are interested in investigating how one can allocate service capacities across storage nodes so as to minimize the download delay when the cumulative service capacity of the system is limited.

Related work

Our work is inspired by a series of recent studies [122, 7, 123, 124, 125, 126, 127, 128, 129, 73], which have analyzed erasure codes (mainly, MDS codes) through the lens of queuing theory, and have shown that the download latency can be improved by employing an erasure coding scheme.

MDS codes and replication schemes have been considered under different aspects of the queuing systems, some of which are pointed out in the following list:

1. Several access schemes, such as fork-join access [122, 126, 127, 125], and centralized *MDS queue* [123, 73];

2. Various service time distributions, namely, exponential [122, 123], shifted exponential [124], *new-worse-than-used* (NWU) and *new-better-than-used* (NBU) distributions [73], more general service time distributions [126];

3. Different traffic patterns and system loads, such as low traffic [122] and high traffic [129], and arbitrary arrival processes [133];

4. Heterogeneous job classes, for instance, MDS coding with heterogeneous jobs classes [134, 128] and replication schemes with heterogeneous job classes [125];

5. Investigation of computing cost [127], job cancellation cost [135], job size variability [136].

However, all these references analyze the download delay when a user is interested in downloading the entire set of files that are jointly encoded using an erasure code. In contrast to such *all-data-access* scenario considered in these references, we are interested in a *hot-data-access* scenario, which has received surprisingly little research attention. In particular, we are concerned with the scenario, in which users are interested in downloading particular popular files (hot data) jointly encoded with other files. We outline our contributions in the following.

Our Contributions

We consider two specific request traffic scenarios: (i) low arrival rate scenario, in which there is no overlap between different download requests, *i.e.*, the current request is served before the arrival of the next request; (ii) high arrival rate scenario, in which the download requests can overlap and local queues are employed to control access to storage nodes.

For low arrival rate scenario, we consider the following two types of content access models. (i) Fork-join access model: wherein a request is forwarded to the systematic node and all its repair groups. The request is said to be complete when either the systematic node completes the request, or all the nodes in a repair group complete requests; (ii) Traffic flow splitting models: certain fraction of download requests would be forwarded to the systematic node, while the remaining fraction would be forwarded to one or more repair groups. Our motivation behind considering such models is that, in practice, it may not always be possible to forward the request to all the nodes. This can happen when, for instance, certain parts of the network are congested or some nodes are temporarily down for maintenance.

Our contributions can be summarized as follows. First, for low-arrival rate scenario, we characterize the mean download latency for the availability codes for the fork-join access in the low arrival rate scenario, and compare it with that of replication and MDS codes.

We demonstrate that, surprisingly, MDS codes perform better in terms of download delay than availability codes. More specifically, we observe that an MDS code results in a lower download delay and a higher fault tolerance (measured in terms of minimum distance) than the availability codes proposed in [17, 22] of the same the code rate (which reflects storage efficiency) for the same cumulative service rate of the system. In addition, we find that replication scheme achieves the same download delay as an MDS code, when the cumulative service rates are the same.

Further, we analyze the impact of the size and the number of multiple repair groups on download latency in low arrival rate scenario by computing the following two metrics: the mean download time achievable by all repair groups working together, and the probability that the repair groups together perform slower download than the systematic node.

Second, for the high arrival rate scenario, we present an upper bound on the mean download time. We also present an inner bound on the stability region – a sufficient condition on the aggregate arrival rate of download requests such that the mean download time is bounded. Finally, we perform simulation studies to compare the download latency of availability codes with that of replication schemes and MDS codes. We observe that when the request arrival rates are low, download delay of availability codes is comparable with that of replication codes, however, for higher arrival rates replication codes have lower latencies. This behavior is analogous to a typical communication system where one uses coding at high SNRs but repetition at low SNRs.

Finally, we consider the problem of service capacity provisioning across storage nodes so as to minimize the file download delay in low arrival rate scenario. We consider three traffic splitting models depending on how the requests are forwarded to the systematic node and to one or more of its repair groups. Under these models, we characterize optimal allocation of service rates across systematic node and repair group nodes to minimize the download delay.

6.2 Preliminaries on Log Concavity (Convexity) of Order Statistics

In this section, we present a result on the log-concavity/log-convexity of the density of a minimum of maximum of i.i.d. random variables. We begin with basic definitions. We refer the reader to [137] for a comprehensive treatment of log-concavity of probability densities.

A function f is said to be log-concave (resp. log-convex) if $\ln f$ is concave (reps. onvex). We say that a random variable S is log-concave (resp. log-convex) if its density function $f_S(s)$ is log-concave (reps. log-convex).

Let $Y = S_{r:r}$ denote the maximum of r i.i.d. random variables each with probability density function $f_S(s)$, whose support is the interval (a, b). Let $X = Y_{1:t}$ be the minimum of t i.i.d. random variables each with probability density function $f_Y(y)$. Then, we have the following results.

Claim 6. *If $f_S(s)$ is continuously differentiable and log-concave on (a, b), then $f_Y(y)$ is also log-concave on (a, b); and if $f_Y(y)$ is continuously differentiable on (a, b), then $f_X(x)$ is also log-concave on (a, b).*

Proof: If $f_S(s)$ is continuously differentiable and log-concave on (a, b), then the cumulative distribution function $F_S(s)$ is also log-concave on (a, b) [137]. Further, we have

$$f_Y(y) = rF_S(s)^{r-1}f_S(s).$$

Observe that $\ln(f_Y(y))$ is a weighted sum of concave functions with non-negative weights, and thus $f_Y(y)$ is also log-concave.

Next, note that since $f_Y(y)$ is log-concave on (a, b), if it is continuously differentiable on (a, b), then the complementary cumulative distribution function $\bar{F}_Y(y)$ is also log-concave on (a, b) [137]. Further, we have

$$f_X(x) = t\bar{F}_Y(y)^{t-1} f_Y(y).$$

Observe that $\ln(f_X(x))$ is a weighted sum of concave functions with non-negative weights, and thus $f_X(x)$ is also log-concave. ■

Claim 7. *If $f_S(s)$ is continuously differentiable and log-convex on (a, b), and if $f_S(a) = 0$, then $f_Y(y)$ is also log-concave on (a, b). If $f_Y(y)$ is continuously differentiable on (a, b), and if $f_Y(b) = 0$, then $f_X(x)$ is also log-convex on (a, b).*

Proof: The proof essentially follows the same lines as the one for log-concavity. The extra condition $f_S(a) = 0$ is required for $F_Y(y)$ to be log-convex, and $f_Y(b) = 0$ is required for $\bar{F}_Y(y)$ to be log-convex (see [137]). ■

6.3 System Model

6.3.1 Encoding Model

We assume that the data to be stored, say $\mathcal{F}$, is divided into k equal sized files. We denote $\mathcal{F} = \{f_1, f_2, \ldots, f_k\}$, where each file f_i is a sequence of symbols that belong to some finite field, and encoding is performed on symbol-by-symbol basis (scalar coding). The files in $\mathcal{F}$ are encoded using a systematic (n, k) code, and the encoded files are stored on n storage nodes. The model can be generalized for vector linear codes.

We are interested in a special sub-class of Locally Repairable Codes (LRC's) for which each systematic node has (r, t)-*availability* [22, 130, 27]. The (r, t)-availability ensures that each systematic node can be regenerated using one of the t disjoint *repair groups* of other storage nodes, each of size at most r (typically, $r \ll k$). We denote such an LRC as an (n, k, r, t)-LRC code. . The size of the repair group r is referred to as the *locality* of the code. We denote the l-th repair group for the i-th node as $\mathcal{R}_i^l$, where $|\mathcal{R}_i^l| \leq r$.

Example: Consider a $(7, 3, 2, 3)$-LRC as follows: $\{f_1, f_2, f_3, f_1 + f_2, f_1 + f_3, f_2 + f_3, f_1 + f_2 + f_3\}$, where files are respectively stored on nodes one to seven. Note that each systematic node can be repaired from three repair groups each of size 2, *e.g.*, $\mathcal{R}_1^1 = \{2, 4\}$, $\mathcal{R}_1^2 = \{3, 5\}$, and $\mathcal{R}_1^3 = \{6, 7\}$.

The (r, t)-availability property allows one to reconstruct a systematic symbol in $(t+1)$ ways: either directly from the systematic node or by decoding of all symbols in one of its t repair groups. For instance, in the above $(7, 3, 2, 3)$-LRC, we can download file f_1 in one of the following ways: (i) download directly from node 1, (ii) download f_2 and $f_1 + f_2$ from nodes 2 and 4, respectively, and decode f_1, or (iii) download f_3 and $f_1 + f_3$ from nodes 3 and 5, respectively, and decode f_1, or (iv) download $f_2 + f_3$ and $f_1 + f_2 + f_3$ from nodes 6 and 7, respectively, and decode f_1.

Throughout, we assume that the time required to reconstruct (decode) a file from the contents of its repair group is negligible in comparison with other delays. For simplicity, we assume that all repair groups are of equal size r.

6.3.2 Content Access Models

Request Arrival Model

We consider *low and high arrival rate* regimes. In the low arrival rate regime, we assume that the system can complete processing the current request before the next request arrives. This assumption serves as a starting point for the latency analysis, and can be useful in systems which are provisioned with enough resources such that the mean service time at a node is significantly smaller than the inter-arrival time of download requests.

In the high arrival rate regime, subsequent requests may arrive before the current request(s) are processed. Hence, there is a need to queue the requests before they can be served. We assume that the requests forwarded to a node are queued at the node in a first-come-first-serve local queue. After a request reaches the head of the queue, it takes certain random service time to read the content from the node. We consider that the arrivals for each file follow an independent Poisson process with a known arrival rate.

Service Model

At node i, the time S_i required to read its contents, referred to as service time, is considered to be a random variable. We assume that S_i is an exponential random variable with parameter μ, denoted as $S_i \sim Exp(\mu)$, and is i.i.d. for each node i.We denote the probability density function (PDF) of S_i as $f_S(s)$ and its cumulative distribution function (CDF) as $F_S(s)$.

Every request is forwarded to the systematic node containing the requested file as well as all its repair groups. Whenever a request is forwarded to a repair group, it is *forked* into r nodes. A repair group is said to complete the download when *all* the r nodes finish their service. The download is complete when either the systematic node or any one of the t repair groups finishes the download.

6.4 Analysis of Download Time for Low Arrival Rate Regime

6.4.1 Mean Download Time

We begin by analyzing the mean download time for codes with (r, t)-availability in the low arrival rate scenario.

Recall that S_i denotes the random variable corresponding to the service time at node i. Let $S^l_{r:r}$ denote the random time required by all the r nodes in its l-th repair group to complete their respective downloads. Notice that $S^l_{r:r}$ would be the maximum of r exponential random variables, *i.e.*,

$$S^l_{r:r} = \max_{j \in \mathcal{R}^{(i)}_l} S_j. \tag{6.1}$$

Let $T^{(r,t)}$ be the random variable that denotes the time required to download a file using a code with (r,t)-availability. Then, we have

$$T^{(r,t)} = \min\left\{S_i, \min_{1\le l\le t} S^l_{r:r}\right\}. \tag{6.2}$$

This is because $T^{(r,t)} = S_i$ if the systematic node downloads its file first. Otherwise, $T^{(r,t)} = S^{l^*}_{r:r}$, if the repair group $\mathcal{R}^{l^*}_i$ completes all its requests before the systematic node.

Theorem 15. *For the low arrival rate model using a code with (r,t)-availability, the mean download time for a file is given as*

$$\mathbb{E}\left[T^{(r,t)}\right] = \frac{1}{\mu r}\beta\left(t+1, \frac{1}{r}\right), \tag{6.3}$$

where $\beta(x,y)$ is the beta function defined as $\beta(x,y) = \int_0^1 u^{x-1}(1-u)^{y-1}\,\mathrm{d}u$.

Proof: To find its expectation, we first compute the complimentary CDF of $T^{(r,t)}$ from the first principles as follows.

$$\begin{aligned}
\Pr\left(T^{(r,t)} > s\right) &\overset{(a)}{=} \Pr\left(\min\left\{S_i, \min_{1\le l\le t} S^l_{r:r}\right\} > s\right)\\
&\overset{(b)}{=} \Pr\left(S_i > s\right)\Pr\left(\min_{1\le l\le t} S^l_{r:r} > s\right)\\
&\overset{(c)}{=} \Pr\left(S_i > s\right)\left[\Pr\left(S^1_{r:r} > s\right)\right]^t\\
&\overset{(d)}{=} \Pr\left(S_i > s\right)\left[\Pr\left(\max_{j\in\mathcal{R}^1_i} S_j > s\right)\right]^t\\
&= \Pr\left(S_i > s\right)\left[1 - \Pr\left(\max_{j\in\mathcal{R}^1_i} S_j \le s\right)\right]^t\\
&\overset{(e)}{=} \Pr\left(S_i > s\right)\left[1 - \left[\Pr\left(S_j \le s\right)\right]^r\right]^t\\
&\overset{(f)}{=} \exp(-\mu s)\left[1 - \left(1 - \exp(-\mu s)\right)^r\right]^t,
\end{aligned} \tag{6.4}$$

where (a) follows from (6.2), (b), (c), and (e) follow from the independence of S_i's, (d) follows from (6.1), and (f) follows since the service times S_i's are exponential with rate μ.

Next, observing that $T^{(r,t)}$ is a non-negative random variable, we have

$$\begin{aligned}&\mathbb{E}\left[T^{(r,t)}\right] = \int_{s=0}^{\infty} \Pr\left(T^{(s,t)} > s\right) \mathrm{d}s \overset{(g)}{=} \\ &\frac{1}{\mu r}\int_{v=0}^{1} v^{t} \ (1-v)^{\left(\frac{1}{r}-1\right)}\mathrm{d}v \overset{(h)}{=} \frac{1}{\mu r}\beta\left(t+1,\frac{1}{r}\right),\end{aligned} \tag{6.5}$$

where (g) can be obtained using (6.4) by substituting $1-(1-\exp(-\mu s))^{r} = v$, and (h) follows from the definition of the beta function. ■

Remark 25. *The above expression allows us to examine the effect of increasing the availability t on the mean download time for a fixed locality r. Suppose Δ denotes the relative decrease in the mean download time by enhancing the availability from (r,t) to $(r,t+1)$. Then, using a simple property of beta function that $\beta(x,y+1) = \frac{y}{x+y}\beta(x,y)$, it is straightforward to verify that*

$$\Delta := \frac{\mathbb{E}\left[T^{(r,t)}\right] - \mathbb{E}\left[T^{(r,t+1)}\right]}{\mathbb{E}\left[T^{(r,t)}\right]} = \frac{1}{r(t+1)+1}. \tag{6.6}$$

For large values of r,t, one can see that $\Delta \approx \frac{1}{rt}$. In other words, if t is large, increasing t further has diminishing returns.

6.4.2 Comparison with Replication Codes

Consider a replication code with replication factor t_r. Such a replication code stores t_r copies of each of the k files, and can be described as a $(n_r = kt_r, k, 1, t_r - 1)$-code. Let the service time of a node (S_i) be an exponential random variable with rate μ_r, i.i.d. across nodes. For the fairness of comparison, we assume that the system using the replication scheme has the same cumulative mean service rate as the system using an (n,k,r,t)-LRC. In other words, we have $n_r\mu_r = n\mu$. Let $T^{(t_r)}$ denote the file download time for a scheme using a t_r-replication code.

Lemma 10. *Consider a t_r-replication code that has the same cumulative mean service rate as that of the system using an (n, k, r, t)-LRC. Then, for the low arrival rate model, the mean download time for a file is given as*

$$\mathbb{E}\left[T^{(t_r)}\right] = \frac{k}{n\mu}. \tag{6.7}$$

Proof: Notice that, for fairness of comparison, we consider that the replication scheme has the same cumulative mean service time as that of the system using an (n, k, r, t)-LRC. Therefore, if μ_r is the mean service time of a node in the replication scheme, then we have $n_r\mu_r = n\mu$, where $n_r = t_r\, k$ is the total number of nodes in the replication scheme. Therefore, we have $\mu_r = \frac{n\mu}{t_r k}$.

Now, in the replication scheme, a request is forked into all the t_r replicas of the systematic node containing the file, and the request is complete when any one of the replicas fetches the file. Thus, the mean download time behaves as the minimum of t_r independent, exponential random variables each with mean $\frac{1}{\mu_r}$. Therefore, we have $\mathbb{E}\left[T^{(t_r)}\right] = \frac{t_r}{\mu_r}$. Substituting the value of μ_r gives the result (6.7). ■

Remark 26. *If we consider $t_r = t + 1$, we get a replication code having the same parameter t as that of an (n, k, r, t)-code. On the other hand, if we consider $t_r = d_{min}$, where d_{min} is the minimum Hamming distance of the (n, k, r, t)-code, we get a replication code that has the same fault-tolerance as that of an (n, k, r, t)-code. Interestingly, as long as the replication code has the same cumulative mean service rate as that of an (n, k, r, t)-LRC, the mean download time of the replication coded system under the low arrival rate model only depends on the rate $\frac{k}{n}$ of the comparable LRC.*

Table 6.1: Normalized mean download time for different coding approaches

No.	Code	$\mathbb{E}[T]\,\mu$	Fault-tolernace
1.	$(30, 15, 3, 2)$-code [22]	$1/1.5556$	7
2.	$(30, 15, 2, 1)$-code [17]	$2/3$	9
3.	$(120, 15, 1, 7)$-replication	0.5	7
4.	$(30, 15)$-MDS	0.5	15

6.4.3 Comparison with MDS Codes

Since MDS codes reduce the mean download time for the entire data (see, *e.g.*, [138, 122, 72]), it is natural to ask how MDS codes would perform in downloading individual files.

Consider a storage system using an (n, k)-MDS code. When a file download request arrives, it is forked into n tasks which are sent to each of the n nodes. A job is complete if the systematic node containing the requested file completes the service or any k out of the remaining $n - 1$ nodes complete their service. Notice that since the code is MDS, all the k files can be reconstructed from the contents of any k nodes, from which, the required file can be obtained. Here as well, we assume that $S_i \sim Exp(\mu)$, and it is i.i.d. across the nodes.

In this case, it is easy to see that the download time $T^{(n,k)}$ of the MDS coded system can be given as

$$T^{(n,k)} = \min\left\{S_i, S_{k:(n-1)}\right\}, \tag{6.8}$$

where $S_{k:(n-1)}$ denotes the k-th order statistics out of $n - 1$ i.i.d. exponential random variables. This allows us to get the following result for the mean download time.

Lemma 11. *For the low arrival rate model using an* (n, k)*-MDS code, the mean download time for a file is given as*

$$\mathbb{E}\left[T^{(n,k)}\right] = \frac{k}{n\mu}. \tag{6.9}$$

Proof: From (6.8), we have $T^{(n,k)} = \min\left\{S_i, S_{k:(n-1)}\right\}$. To find the expectation, we first find the complementary CDF as follows.

$$\begin{aligned}
&\Pr\left(T^{(n,k)} > s\right) \\
&= \Pr\left(\min\left\{S_i, S_{k:(n-1)}\right\} > s\right) \\
&\overset{(h)}{=} \Pr\left(S_i > s\right)\Pr\left(S_{k:(n-1)} > s\right) \\
&\overset{(i)}{=} \Pr\left(S_i > s\right)\left\{\sum_{j=0}^{k-1}\binom{n-1}{j}\left[\Pr\left(S_i \le s\right)\right]^j\left[\Pr\left(S_i > s\right)\right]^{n-j-1}\right\} \\
&\overset{(j)}{=} exp(-\mu s)\left\{\sum_{j=0}^{k-1}\binom{n-1}{j}(1 - exp(-\mu s))^j\, exp(-\mu(n-j-1)s)\right\} \\
&= \sum_{j=0}^{k-1}\binom{n-1}{j}\exp\left(-\mu(n-j)s\right)\left(1-\exp(-\mu s)\right)^j, \qquad (6.10)
\end{aligned}$$

where (h) follows from the independence of S_i's. For obtaining (i), we use the standard expression for the CDF of the k-th order statistics out of $n-1$ independent random variables as $\Pr\left(S_{k:(n-1)} \le s\right) = \sum_{j=k}^{n-1}\binom{n-1}{j}\left[\Pr\left(S_i \le s\right)\right]^{(j)}\left[\Pr\left(S_i > s\right)\right]^{(n-j-1)}$, which gives $\Pr\left(S_{k:(n-1)} > s\right) = 1 - \sum_{j=k}^{n-1}\binom{n-1}{j}\left[\Pr\left(S_i \le s\right)\right]^{(j)}\left[\Pr\left(S_i > s\right)\right]^{(n-j-1)} = \sum_{j=0}^{k-1}\binom{n-1}{j}\left[\Pr\left(S_i \le s\right)\right]^{(j)}\left[\Pr\left(S_i > s\right)\right]^{(n-j-1)}$. Finally, (j) follows from the fact that each $S_i \sim \exp(\mu)$.

Now, noting that $T^{(n,k)}$ is a non-negative random variable, we have

$$\begin{aligned}
&\mathbb{E}\left[T^{(n,k)}\right] \\
&= \int_{s=0}^{\infty} \Pr\left(T^{(n,k)} > s\right) \mathrm{d}s \\
&\overset{(k)}{=} \int_{s=0}^{\infty} \sum_{j=0}^{k-1} \binom{n-1}{j} \exp\left(-\mu(n-j)s\right)\left(1-\exp(-\mu s)\right)^{j} \mathrm{d}s \\
&\overset{(l)}{=} \sum_{j=0}^{k-1} \binom{n-1}{j} \int_{s=0}^{\infty} \exp\left(-\mu(n-j)s\right)\left(1-\exp(-\mu s)\right)^{(j)} \mathrm{d}s \\
&\overset{(m)}{=} \sum_{j=0}^{k-1} \binom{n-1}{j} \frac{1}{\mu} \int_{v=0}^{1} v^{j}(1-v)^{n-j-1} \mathrm{d}v \\
&\overset{(n)}{=} \frac{1}{\mu} \sum_{j=0}^{k-1} \binom{n-1}{j} \beta(j+1, n-j) \\
&\overset{(o)}{=} \frac{k}{n\mu},
\end{aligned} \tag{6.11}$$

where, (k) follows by substituting (6.10), (l) follows from changing the order of summation and integration, (m) follows from change of variables $v = (1 - exp(-\mu s))$, and (n) follows from the definition of $\beta(x, y)$ function. Finally, (o) is obtained by using $\beta(x, y) = \frac{(x-1)!\,(y-1)!}{(x+y-1)!}$, when x and y are positive integers. ■

Remark 27. *Under the low arrival rate model, an (n, k)-MDS code has the same mean download time as a t_r-replication code when the cumulative mean service time for the replication system is equal to that of the MDS coded system.*

6.4.4 Discussion

In table I, we compare the mean download time for various codes along with their fault-tolerance capability (fault-tolerance is measured in number of failures that can be sustained without any data loss). The rate of each code gives an indication of its storage efficiency. We consider replication scheme having the same fault-tolerance as that of the $(30, 15, 3, 2)$-

LRC[2]. We observe that MDS code performs the best in terms of all these parameters. One caveat is that, for any MDS code, computational complexity of reconstructing the file would be the highest.

6.4.5 Download Performance of Repair Groups

In this section, we analyze the impact of the number of repair groups t and the size of repair groups r on the download time when each of the repair group nodes is given a fixed service rate, say μ. We do so by studying the latency performance of repair groups with respect to the systematic node. We first compute the expected download time when the request is forked to all the t repair groups. Next, we compute the probability that systematic node is faster in download than its repair groups.

Let $S^{(r,t)}$ be the random variable that denotes the download time for t repair groups each of size r. Then, we have

$$S^{(r,t)} = \min_{1 \leq l \leq t} S^l_{r:r}. \tag{6.12}$$

Expected Download Time of Repair Groups

Proposition 6. *The mean time to download a file using t repair groups each of size r is given as*

$$\mathbb{E}\left[S^{(r,t)}\right] = \frac{1}{\mu} \sum_{m=1}^{t} \binom{t}{m} (-1)^{j-1} H_{rm}, \tag{6.13}$$

where H_{rm} is (rm)- th Harmonic number defined as $H_{rm} = \sum_{j=1}^{rm} \frac{1}{j}$.

[2] The fault-tolerances for the LRC's are obtained from their minimum distance properties mentioned in respective references.

Proof: We first compute the complementary CDF of $S^{(r,t)}$ as follows.

$$\begin{aligned}\Pr\left(S^{(r,t)} > s\right) &= \Pr\left(\min_{1\le l\le t} S^l_{r:r} > s\right)\\ &\overset{(a)}{=} \left[\Pr\left(S^l_{r:r} > s\right)\right]^t\\ &\overset{(b)}{=} \left[1-\left(\Pr\left(S_j \le s\right)\right)^r\right]^t\\ &\overset{(c)}{=} \left[1-\left(1-\exp(-\mu s)\right)^r\right]^t, \qquad (6.14)\end{aligned}$$

where (a) and (b) follow since service times are independent across nodes, and (c) follows since service time at each node is exponentially distributed with parameter μ. Next, since $S^{(r,t)}$ is a non-negative random variable, we have

$$\begin{aligned}\mathbb{E}\left[S^{(r,t)}\right] &= \int_{s=0}^{\infty} \Pr\left(S^{(r,t)} > s\right) \mathrm{d}s\\ &\overset{(d)}{=} \int_{s=0}^{\infty} \left[1-\left(1-\exp(-\mu s)\right)^r\right]^t \mathrm{d}s \qquad (6.15)\\ &\overset{(e)}{=} \int_{s=0}^{\infty} \left(\sum_{j=0}^{t}\binom{t}{j}(-1)^j\right.\\ &\qquad \left.\left(\sum_{l=0}^{rj}\binom{rj}{l}(-1)^l\exp(-\mu s l)\right)\right)\mathrm{d}s\\ &\overset{(f)}{=} \int_{s=0}^{\infty} \left(\sum_{j=1}^{t}\binom{t}{j}(-1)^j\right.\\ &\qquad \left.\left(\sum_{l=1}^{rj}\binom{rj}{l}(-1)^l\exp(-\mu s l)\right)\right)\mathrm{d}s\\ &\overset{(g)}{=} \sum_{j=1}^{t}\binom{t}{j}(-1)^j\left(\sum_{l=1}^{rj}\binom{rj}{l}(-1)^l\frac{1}{\mu l}\right)\\ &\overset{(h)}{=} \frac{1}{\mu}\sum_{j=1}^{t}\binom{t}{j}(-1)^{j-1}H_{rj}, \qquad (6.16)\end{aligned}$$

where (d) can be obtained using (6.14), (e) is obtained by binomial expansions with respect to powers t and r, (f) is obtained by gathering together the terms in inner summation for $l =$

0 and using $\sum_{j=0}^{t}\binom{t}{j}(-1)^j = 0$, (g) is obtaineed by interchanging the order of summation and integration, and (h) is obtained using an identity $H_m = \sum_{l=1}^{m}\binom{m}{l}(-1)^{l-1}\frac{1}{l}$. ■

To get some more insight into how the expected download time of the repair groups behaves with respect to r and t , we derive lower and upper bounds in the following.

Claim 8. *For any natural numbers $r, t > 0$, we have*

$$\frac{1}{\mu}\cdot\frac{H_r}{t} \leq \mathbb{E}\left[S^{(r,t)}\right] \leq \frac{1}{\mu}\cdot H_r\frac{1}{r}\beta\left(t,\frac{1}{r}\right), \tag{6.17}$$

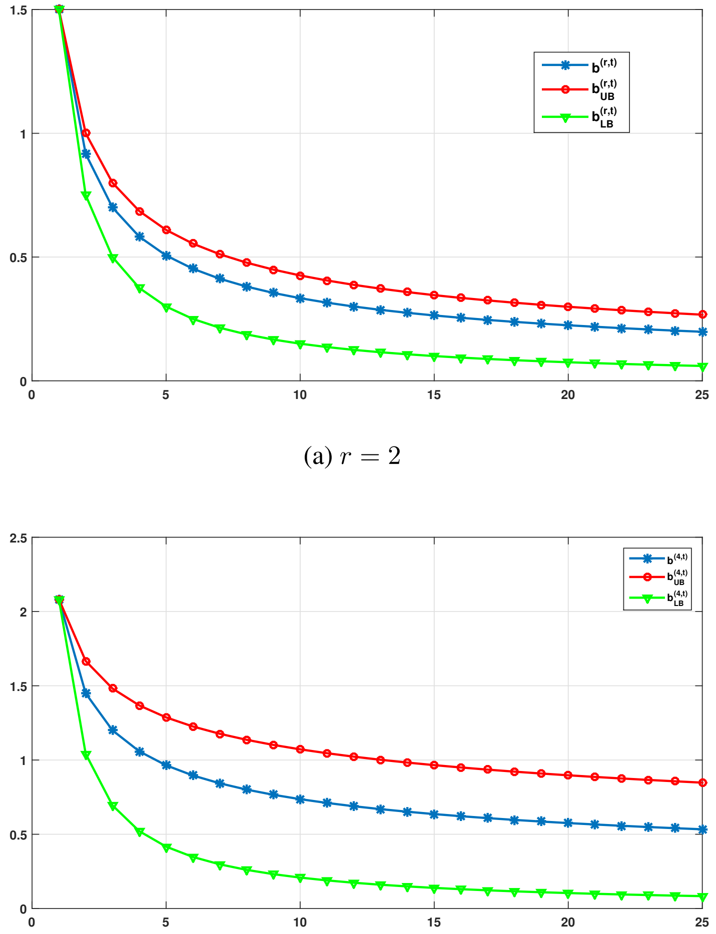
where $\beta(x,y)$ denotes the beta function defined as $\beta(x,y) = \int_{v=0}^{1} v^{x-1}(1-v)^{y-1}dv$.

Proof: To get the upper bound, from (6.15), we have

$$\begin{aligned}\mathbb{E}\left[S^{(r,t)}\right] &= \int_{s=0}^{\infty}\left[1-(1-\exp(-\mu s))^r\right]^t \mathrm{d}s\\ &\overset{(i)}{=} \frac{1}{\mu}\int_{u=0}^{1}\left(\frac{1-u^r}{1-u}\right)(1-u^r)^{t-1}\mathrm{d}u\\ &\overset{(j)}{\leq} \frac{1}{\mu}\int_{u=0}^{1}\left(\frac{1-u^r}{1-u}\right)\mathrm{d}u\int_{u=0}^{1}(1-u^r)^{t-1}\mathrm{d}u\\ &\overset{(k)}{=} \frac{1}{\mu}\cdot H_r\cdot\frac{1}{r}\beta\left(t,\frac{1}{r}\right),\end{aligned} \tag{6.18}$$

where (i) follows from substituting $1-\exp(-\mu s) = u$. To obtain (j), we use Harris's inequality [139, Theorem 2.15], which says that, for a random variable x, if $g(x)$ is a non-decreasing function and $h(x)$ is a non-increasing function, then $\mathbb{E}[g(x)h(x)] \leq \mathbb{E}[g(x)]\,\mathbb{E}[h(x)]$. We consider the random variable u distributed uniformly over $[0,1]$, and functions $g(u) = \frac{1-u^r}{1-u}$ and $h(u) = (1-u^r)^{t-1}$. To obtain (k), we use the integral representation of a harmonic number given as $H_r = \int_{u=0}^{1}\left(\frac{1-u^r}{1-u}\right)\mathrm{d}u$, and we obtain $\int_{u=0}^{1}(1-u^r)^{t-1}\mathrm{d}u = \frac{1}{r}\beta\left(t,\frac{1}{r}\right)$ by substituting $1-u^r = v$.

To get the lower bound, we observe that, being an exponential random variable, S_i is a *log-concave* random variable. (See Sec. 6.2 for discussion on log-concavity and log-



(a) $r = 2$

(b) $r = 4$

Figure 6.1: Expected service time and its upper and lower bounds as a function of t for various values of r for $\mu = 1$.

convexity of random variables.) This allows us to show that $S^l_{r:r}$ is also log-concave. (Sec. 6.2, Claim 6.) Then, we use a result in [126] that for a log-concave random variable S with PDF $f_S(s)$, we have $\mathbb{E}\left[S\right] \leq t\mathbb{E}\left[S_{1:t}\right]$, where $S_{1:t}$ is the minimum of t i.i.d. random variables each with PDF $f_S(s)$. Notice that using $S = S^l_{r:r}$ results in $S_{1:t} = S^{(r,t)}$, and it is well known order-statistics result that the expected value of the maximum of exponential random variables is $\mathbb{E}\left[S^l_{r:r}\right] = \frac{H_r}{\mu}$ (see [140]). ■

The equalities hold for all $t \geq 1$ when $r = 1$. In Fig. 6.1, we numerically evaluate the tightness of the bound in (6.17). Observe that the bounds become loose as the locality r increases.

Remark 28. *Suppose we are using an (n, k)-MDS code to store the files. Consider fork-join type of policy for MDS codes, wherein a download request is forked to all the n nodes. The request is complete if the systematic node containing the file completes its service or if any k out of the remaining $n - 1$ nodes complete their service. In this case, the expected download time using parity nodes is the k-th order statistic among $n - 1$ i.i.d. exponential RVs. Therefore, using the well-known result on mean of k-th order-statistics of exponential random variables (see [140]), we have*

$$\mathbb{E}\left[T_{MDS}^{(n-1,k)}\right] = \frac{1}{\mu}\left(H_{n-1} - H_{n-1-k}\right). \tag{6.19}$$

Probability that Systematic Node is Faster than its Repair Groups

We analyze the probability that all the repair groups together take more time to complete the service than the systematic node.

Proposition 7. *The probability that repair groups are slower than the systematic node is given as*

$$\Pr\left(T_{FJ}^{(r,t)} > S_i\right) = \frac{1}{r}\beta\left(t+1, \frac{1}{r}\right), \tag{6.20}$$

where $\beta(x, y)$ denotes the beta function defined as $\beta(x, y) = \int_{v=0}^{1} v^{x-1}(1 - v)^{y-1} dv$.

Proof:

$$\begin{aligned}
\Pr\left(S^{(r,t)} > S_i\right) &= \int_{s=0}^{\infty} \Pr\left(S^{(r,t)} > s\right) f_S(s)\mathrm{d}s \\
&\overset{(a)}{=} \int_{s=0}^{\infty} \left[1 - \left(\Pr\left(S_j \leq s\right)\right)^r\right]^t f_S(s)\mathrm{d}s \\
&\overset{(b)}{=} \int_{s=0}^{\infty} \left[1 - \left(1 - \exp(-\mu s)\right)^r\right]^t \\
&\qquad \mu \exp(-\mu s)\mathrm{d}s \\
&\overset{(c)}{=} \frac{1}{r}\beta\left(t+1, \frac{1}{r}\right),
\end{aligned}$$

where (a) follows from (6.14), (b) follows since $S_i \sim Exp(\mu)\ \forall\ i$, and (c) follows from substituting $1 - (1 - \exp(-\mu s))^r = v$. ■

Notice that, since t is a non-negative integer, one can write

$$\frac{1}{r}\beta\left(t+1, \frac{1}{r}\right) = \frac{t!}{\left(\frac{1}{r}+1\right)\left(\frac{1}{r}+2\right)\cdots\left(\frac{1}{r}+t\right)}.$$

This clearly shows that $0 \leq \frac{1}{r}\beta\left(t+1, \frac{1}{r}\right) \leq 1$. Further, one can observe that this probability expression increases with r and decreases with t, which corroborates the intuition.

We plot the probability versus t for various values of r when $\mu = 1$ in Fig. 6.2. Observe that increasing r greatly increases the probability that repair groups perform slower than systematic node. For instance, if we increase r from 1 to 3, we need to increase t from 1 to 5 to keep the probability of success the same (*i.e.*, 0.5).

Remark 29. *Similar to above, we can compute the probability that the systematic node is faster than parity nodes in an (n, k)-MDS code. Recall the fork-join type of policy for MDS codes, wherein a download request is forked to all the n nodes. The request is complete if the systematic node containing the file completes its service or if any k out of $n-1$ remaining nodes complete their service.*

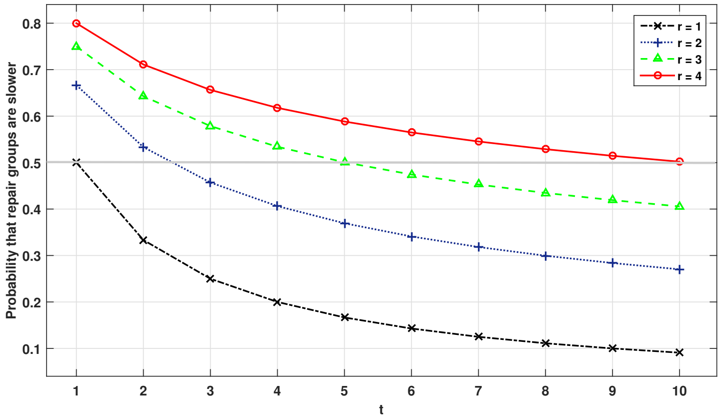


Figure 6.2: Probability that the repair groups download faster than the systematic node as a function of t for various values of r when $\mu = 1$.

Proposition 8. *For an (n, k)-MDS code, the probability that parity nodes finish after the systematic node is given as*

$$\Pr\left(T_{MDS}^{(n-1,k)} > S_i\right) = \frac{k}{n}. \tag{6.21}$$

Proof:

$$\begin{aligned}\Pr\left(T_{\text{MDS}}^{(k,n-1)} > S_i\right) &= \int_{s=0}^{\infty} \Pr\left(T_{\text{MDS}}^{(k,n-1)} > s\right) f_S(s)\mathrm{d}s \\ &\overset{(d)}{=} \int_{s=0}^{\infty} \Pr\left(S_{k:n-1} > s\right) f_S(s)\mathrm{d}s \\ &\overset{(e)}{=} \int_{s=0}^{\infty} \sum_{j=0}^{k-1} \binom{n-1}{j} \\ &\quad F_S(s)^j(1-F_S(s))^{n-1-j} f_S(s)\mathrm{d}s\end{aligned}$$

where (d) follows from the fact that the request is complete when any k out of $n-1$ nodes finish their service, (e) follows from $F_{S_{k:n}}(s) = \sum_{j=k}^{n} \binom{n}{j} F_S(s)^j (1 - F_S(s))^{n-j}$. Using the fact that service time at each node is an exponential random variable, we get

$$\Pr\left(T_{\text{MDS}}^{(k,n-1)} > S_i\right) = \int_{s=0}^{\infty} \sum_{j=0}^{k-1} \binom{n-1}{j} (1-\exp(-\mu s))^j \\ \exp(-\mu(n-1-j)s)\Big)\mu\exp(-\mu s)\mathrm{d}s$$

After substituting $\exp(-\mu s) = v$ and rearranging the terms, we get

$$\begin{aligned}\Pr\left(T_{\text{MDS}}^{(k,n-1)} > S_i\right) &= \sum_{j=0}^{k-1} \binom{n-1}{j} \int_{v=0}^{1} (1-v)^j v^{n-j-1}\mathrm{d}v \\ &= \sum_{j=0}^{k-1} \binom{n-1}{j} \beta(j+1, n-j) \\ &= \sum_{j=0}^{k-1} \frac{(n-1)!}{j!\,(n-1-j)!} \cdot \frac{j!\,(n-j-1)!}{n!} \\ &= \frac{k}{n}. \end{aligned} \tag{6.22}$$

■

6.5 Results for the High Arrival Rate Scenario

In this scenario we assume that one or more requests can arrive before the current event is served. As before, we assume that the service time at each node is an i.i.d. exponential random variable with mean $\frac{1}{\mu}$.

We consider that the download requests arrive as a Poisson process with aggregate rate λ. Requests are split according to the *popularity* of the files. In particular, a file f_i is said to have the popularity p_i such that $0 \leq p_i \leq 1$ for each i and $\sum_{i=1}^{k} p_i = 1$. Notice that the download requests for a file f_i form a Poisson process with arrival rate λp_i.

There are two levels of fork-join (FJ) queues[3] present in the system. On the outer side,

[3] In an (n, k)-FJ queue, where $n \geq k$, a job is forked into n tasks. The job is complete, once any k out of the n tasks are complete. The remaining $n - k$ tasks are immediately *killed* (removed from the system). See [72] for details on (n, k)-FJ queues.

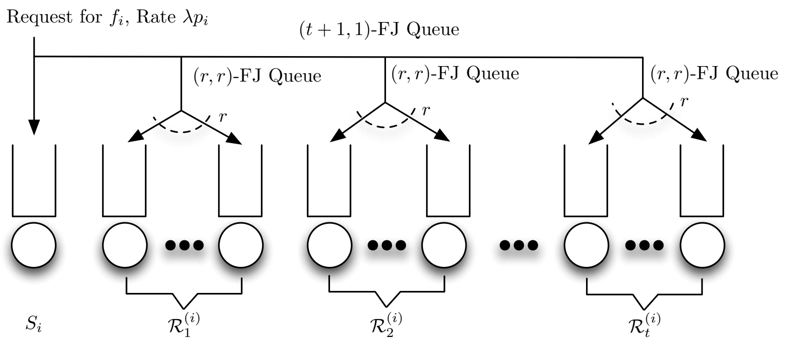


Figure 6.3: Fork-join queueing system with an (n,k,r,t)-LRC.

the $(t+1)$ tasks form a $(t+1,1)$-FJ queue. On the inner side, the r sub-tasks associated with each of the t repair groups form an (r,r)-FJ queue. (See Fig. 6.3.)

Example: Consider a simple $(3,2,2,1)$-LRC code $\{f_1, f_2, f_1 + f_2\}$. We call the FJ queue for this particular code as the *butterfly queue*, since the code resembles the one used over the butterfly network for network coding. (See Fig. 6.4.) Requests for file f_1 form a Poisson process with rate λp_1, while those for f_2 form a Poisson process with rate λp_2 such that $p_1, p_2 \geq 0$, $p_1 + p_2 = 1$. Each job is forked into three (sub)tasks. Request for a file f_1 (resp. f_2) is complete if either f_1 (resp. f_2) is downloaded directly or both f_2 and $f_1 + f_2$ (resp. both f_1 and $f_1 + f_2$) are downloaded. If the systematic task completes first, the remaining sub-tasks corresponding to the request are removed from the system. The systematic task is removed from the system if the other two sub-tasks complete before it.

6.5.1 Upper Bound on Mean Download Time

Fork-join queues are difficult to analyze. Only bounds have been known for the mean download time except for a $(2,2)$ FJ queue (see *e.g.*, [141, 142]). Therefore, we obtain bounds on the performance of the FJ queueing system by considering a more restricted queueing model, known as the *split-merge* (SM) system. We note that the SM system is

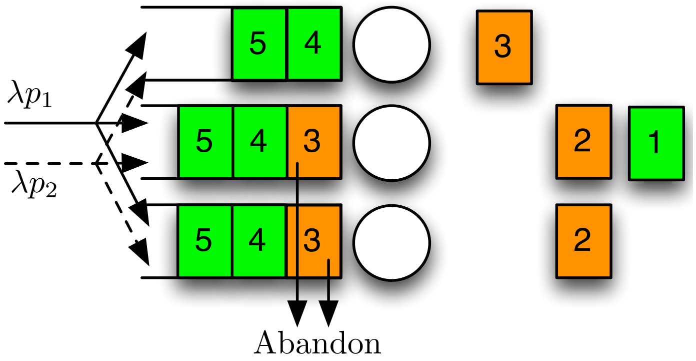


Figure 6.4: Butterfly Queue: Requests for f_1 are shown in green whereas those for f_2 are shown in brown.

also used in [122, 72] for characterizing upper bounds on the (n, k)-FJ queue.

In the FJ system, a node can service the tasks of the next job in its local queue once it completes the current task. This creates dependencies in waiting times across multiple nodes, and the service time of a job becomes dependent on the departure time of the previous job. On the other hand, in the SM system, all the n nodes are blocked until the job is complete. Since the nodes are blocked in the SM system, its mean download time gives an upper bound on the download time of the FJ system.

Theorem 16. *For a fork-join queueing system using an (n, k, r, t)-LRC, an upper bound on the mean download time of a file is given by*

$$\mathbb{E}\left[T_{FJ}^{(r,t)}\right] \leq \frac{1}{\mu r}\beta\left(t+1,\frac{1}{r}\right) + \frac{r\lambda \sum_{j=0}^{t}\binom{t}{j}(-1)^j\sum_{l=0}^{rj}(-1)^l\binom{rj}{l}\frac{1}{(l+1)^2}}{\left(r\mu^2 - \lambda\mu\beta\left(t+1,\frac{1}{r}\right)\right)}. \tag{6.23}$$

Here $\beta(x, y)$ is the beta function defined as $\beta(x, y) = \int_0^1 u^{x-1}(1-u)^{y-1}\,\mathrm{d}u$. The bound is valid only when $\lambda < \frac{\mu r}{\beta\left(t+1,\frac{1}{r}\right)}$.

Proof: To obtain an upper bound, we consider the SM queueing model with the same parameters. Notice that the SM queueing model is equivalent to an M/G/1 queue with Poisson arrivals having aggregate arrival rate λ, and the generalized service time equal to that of the low arrival rate model $T^{(r,t)}$ (see (6.3)).

Now, for an M/G/1 queue, the Pollaczek-Khinichin formula [143] allows us to compute the mean download time in terms of the first and second moments of the generalized service time as follows:

$$\mathbb{E}\left[T\right] = \mathbb{E}\left[T\right] + \frac{\lambda \mathbb{E}\left[T^2\right]}{2\left(1 - \lambda \mathbb{E}\left[T\right]\right)}. \tag{6.24}$$

We find the second moment of $T^{(r,t)}$ as follows:

$$\begin{aligned}
\mathbb{E}\left[(T^{(r,t)})^2\right] &\overset{(l)}{=} \int_{s=0}^{\infty} 2s \Pr\left(T^{(r,t)} > s\right) \mathrm{d}s \\
&\overset{(m)}{=} \int_{s=0}^{\infty} 2s\ \exp(-\mu s) \left[1 - (1 - \exp(-\mu s))^r\right]^t \mathrm{d}s \\
&\overset{(n)}{=} \sum_{j=0}^{t} \binom{t}{j} (-1)^j \sum_{l=0}^{rj} (-1)^l \binom{rj}{l} \\
&\quad \times \int_{s=0}^{\infty} 2s\, exp\left(-\mu(l+1)s\right) \mathrm{d}s \\
&= \sum_{j=0}^{t} \binom{t}{j} (-1)^j \sum_{l=0}^{rj} (-1)^l \binom{rj}{l} \frac{2}{\mu^2 (l+1)^2},
\end{aligned} \tag{6.25}$$

where (l) follows since $(T^{(r,t)})^2$ is a non-negative random variable, (m) follows from (6.4), and (n) follows from the binomial expansion of $\left[1 - (1 - \exp(-\mu s))^r\right]^t$ and interchanging the order of integration and summation. Finally, substituting (6.3) and (6.25) into (6.24) gives (6.23). ■

Remark 30. *Since the SM system considers a single centralized queue, the above bound does not capture the dependency of download latency on the popularities of files. More-*

over, from the simulation studies, we note that this bound is fairly loose for higher arrival rates.

6.5.2 Inner Bound on Stability Region

Considering a split-merge system also allows us to characterize an inner bound on the stability region of the FJ system.

Theorem 17. *For a fork-join system with an (n, k, r, t)-LRC, an inner bound on the on the stability region is given as*

$$\lambda < \frac{\mu r}{\beta\left(t+1, \frac{1}{r}\right)}, \tag{6.26}$$

where $\beta(x, y)$ is the beta function defined as $\beta(x, y) = \int_0^1 u^{x-1}(1-u)^{y-1}\,\mathrm{d}u$.

Proof: Since the download time under FJ model is upper bounded by the download time under SM model with same parameters, the stability region of the split-merge model gives an inner bound on the stability region of the fork-join model. Now, as the split-merge system is equivalent to an M/G/1 queue with generalized service time given as $T^{(r,t)}$, its stability condition is $\lambda < \frac{1}{\mathbb{E}\left[T^{(r,t)}\right]}$. The result follows from (6.3). ■

Remark 31. *The above condition gives a sufficient condition for the FJ system to be stable, however, it is not a necessary condition. In the simulation studies, we note that this bound is somewhat loose.*

6.5.3 Simulation Results

In this section, we simulate the fork-join queueing model in the high arrival rate scenario for the codes given in Table I: a $(30, 15, 3, 2)$-LRC in [22], a $(30, 15, 2, 1)$-LRC in [17], an 8-replication scheme that is effectively a $(120, 15, 1, 7)$-LRC, and a $(30, 15)$-MDS code.

In Fig. 6.5, we consider the case when all the files have the same popularity, *i.e.*, $p_i = \frac{1}{15}, \forall\, i \in \{1, \cdots, 15\}$. Observe that the replication scheme gives the lowest latency

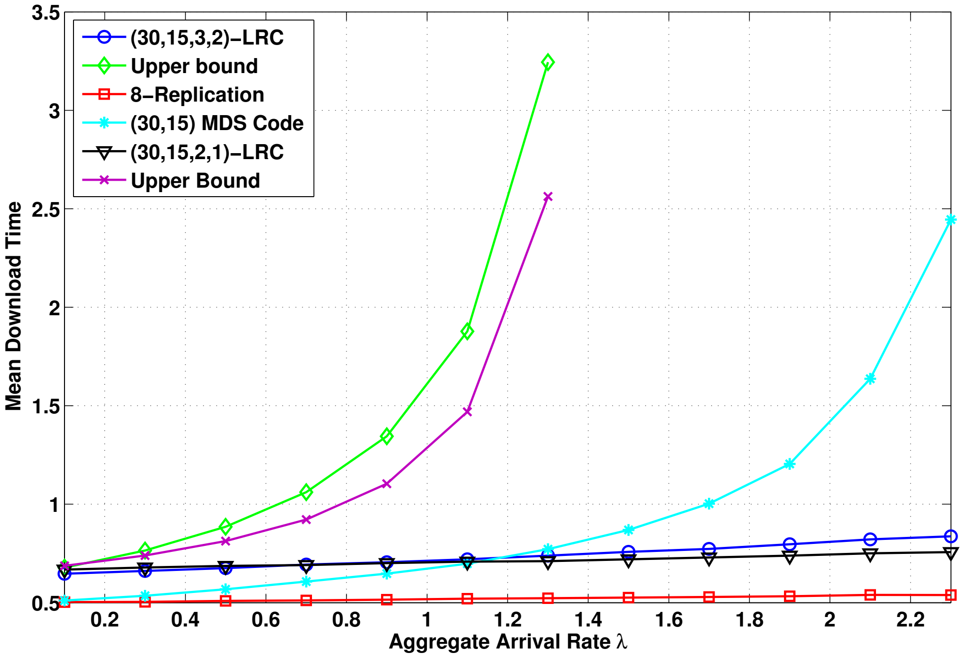


Figure 6.5: Mean download time vs. aggregate arrival rate for various coding schemes when all the files have equal popularity.

at the cost of heavy storage overhead. Both the LRC's have nearly the same latency performance, which is slightly higher than that of the replication. This indicates that, even though increasing the availability parameter t increases the the parallel reads, it may not reduce the download time of a file. The latency for the MDS code increases rapidly with the aggregate arrival rate. This is because, for an MDS code, a request is forked into all the nodes, which increases the arrival rate for the local queues at each node. This tends to saturate the local queues for higher values of aggregate arrival rate. The results show that the availability codes are favorable at higher arrival rates.

Next, we consider the case of unequal popularity of files, wherein 6.67% of the files have 90% of the popularity while the rest of the files equally share 10% popularity (see

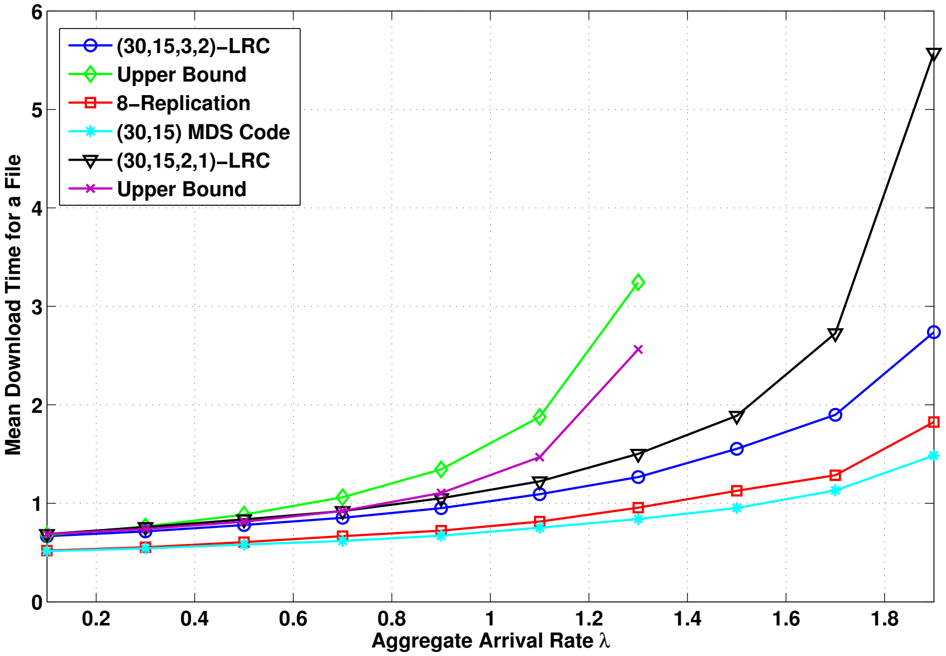
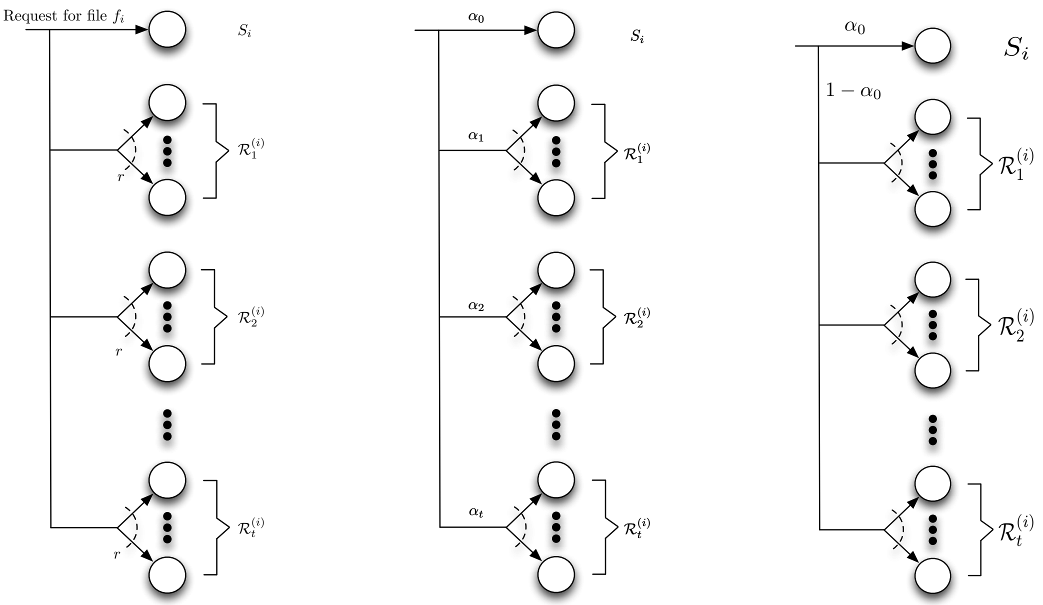


Figure 6.6: Mean download time vs. aggregate arrival rate for various coding schemes when 6.67% of the files have 90% of the popularity and the rest share 10% of popularity equally.

Fig. 6.6). In this case, the latency performance of the replication scheme degrades, while that of the MDS code improves. This is because, in the replication scheme, the local queues at the nodes storing the replica of the most popular file (f_1) get overloaded; whereas, in the case of MDS code, the load of downloading f_1 gets divided into all the nodes. Observe that higher value of t helps in this case, as the $(30, 15, 3, 2)$-LRC performs better than the $(30, 15, 2, 1)$-LRC. From Figures 6.5 and 6.6, we can see that the delay performance of availability codes is fairly robust against sharp changes in popularity.

This also opens up an interesting problem to investigate if one can trade-off t for r for different arrival patterns for obtaining better download delay. Further, it is interesting to compute tighter bounds that also take into account file popularities. A challenging task



(a) No – split (b) $t+1$ – split (c) 2 – split

Figure 6.7: Three traffic flow splitting models.

would be to consider a joint design of storage code and scheduling policy.

6.6 Service Capacity Allocation

In this section, we consider the problem of optimal service capacity provisioning across nodes for availability codes. In particular, given a total budget of $\tilde{\mu}$ units on the cumulative service rate of the system, we allocate $1-\gamma$ fraction to the systematic node $(0 \leq \gamma \leq 1)$, and allocate the rest γ fraction equally among the repair groups. In other words, average service rate of a node is $\mu_i = (1-\gamma)\tilde{\mu}$, for the systematic node i, and $\mu_j = \frac{\gamma}{rt}\tilde{\mu}$, for a node j in a repair group of i. Our goal is to find optimal γ which would minimize the mean download delay. We consider three traffic flow models: no-split, $(t+1)$-split, and 2-split (see Fig. 6.7).

6.6.1 Service Rate Allocation for No-Split Model

Recall that in the no-split (or 0-split) model, a request is forwarded to the systematic node as well as all its repair groups. The request is said to be complete if either systematic node finishes its service or all the r nodes in any one of the t repair groups complete their service. We compute the mean download time as a function of γ in the following.

Proposition 9. *When the systematic node is allocated $1-\gamma$ fraction of the total service rate $\tilde{\mu}$ and the remaining rate $\gamma\tilde{\mu}$ is equally distributed among the repair group nodes $(0 \leq \gamma \leq 1)$, the mean download delay for an (r,t)-availability code using a no-split fork join policy is*

$$\begin{aligned}\mathbb{E}\left[T_{0,FJ}^{(r,t)}(\gamma)\right] = \frac{1}{\tilde{\mu}}\sum_{j=1}^{t}\binom{t}{j}(-1)^{j} \\ \left(\sum_{l=1}^{rj}\binom{rj}{l}(-1)^{l}\frac{1}{1+\gamma\left(\frac{l}{rt}-1\right)}\right). \qquad (6.27)\end{aligned}$$

Proof: First, note that the complementary CDF of $T_{0,\text{FJ}}^{(r,t)}$ can be written as

$$\begin{aligned}\Pr\left(T_{0,\text{FJ}}^{(r,t)} > s\right) &= \Pr\left(\min\left\{S_i, \min_{1\leq l\leq t} S_{r:r}^{l}\right\} > s\right) \\ &\overset{(a)}{=} \exp\left(-(1-\gamma)\tilde{\mu}s\right) \\ &\quad \left[1-\left(1-\exp(-\frac{\gamma\tilde{\mu}}{rt}s)\right)^{r}\right]^{t}, \qquad (6.28)\end{aligned}$$

where (a) follows from the independence of service times and the allocation strategy. Next, we have

$$\begin{aligned}\mathbb{E}\left[T_{0,\text{FJ}}^{(r,t)}\right] &= \int_{s=0}^{\infty} \Pr\left(T_{0,\text{FJ}}^{(r,t)} > s\right) \mathrm{d}s \\ &\overset{(b)}{=} \int_{s=0}^{\infty} \exp\left(-(1-\gamma)\tilde{\mu}s\right) \\ &\qquad \left[1 - \left(1 - \exp\left(-\frac{\gamma\tilde{\mu}}{rt}s\right)\right)^r\right]^t \mathrm{d}s,\end{aligned}$$

where (b) follows from (6.28). After expanding the binomials, we get

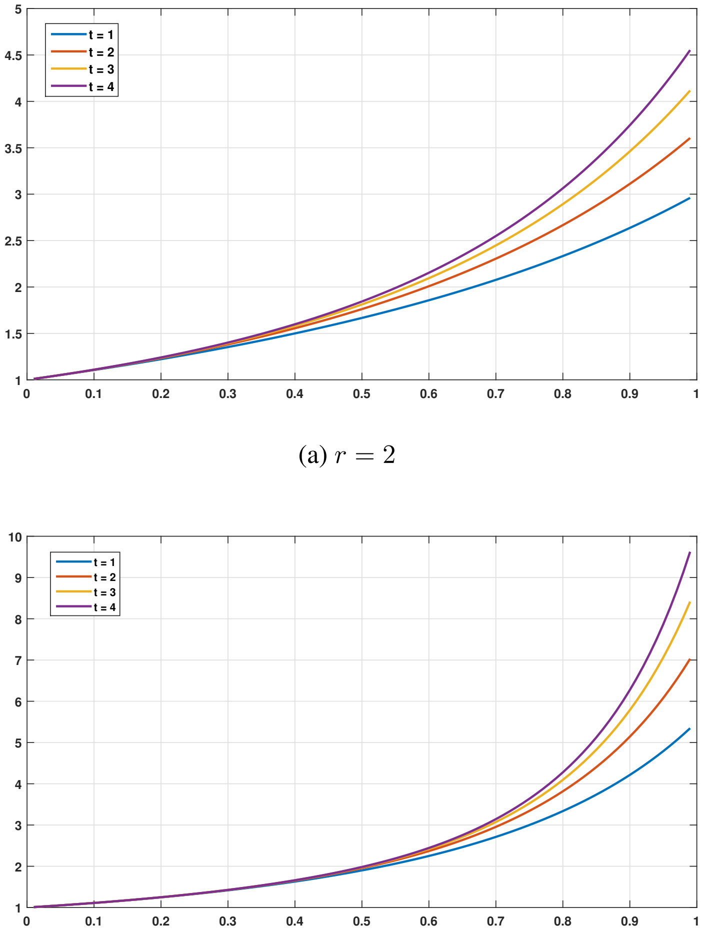
$$\begin{aligned}\Pr\left(T_{0,\text{FJ}}^{(r,t)} > s\right) &= \int_{s=0}^{\infty} \exp\left(-(1-\gamma)\tilde{\mu}s\right) \left(\sum_{j=0}^{t} \binom{t}{j} (-1)^j\right. \\ &\qquad \left.\left(\sum_{l=0}^{rj} \binom{rj}{l} (-1)^l \exp\left(-\frac{\gamma\tilde{\mu}}{rt} sl\right)\right)\right) \mathrm{d}s \\ &\overset{(c)}{=} \int_{s=0}^{\infty} \exp\left(-(1-\gamma)\tilde{\mu}s\right) \left(\sum_{j=1}^{t} \binom{t}{j} (-1)^j\right. \\ &\qquad \left.\left(\sum_{l=1}^{rj} \binom{rj}{l} (-1)^l \exp\left(-\frac{\gamma\tilde{\mu}}{rt} sl\right)\right)\right) \mathrm{d}s \\ &\overset{(d)}{=} \frac{1}{\tilde{\mu}} \sum_{j=1}^{t} \binom{t}{j} (-1)^j \sum_{l=1}^{rj} \binom{rj}{l} \frac{(-1)^l}{1 - \gamma + \frac{\gamma l}{rt}}\end{aligned}$$

where (c) follows from gathering terms for $l = 0$ and using $\sum_{j=0}^{t} \binom{t}{j} (-1)^j = 0$, and (d) follows from interchanging the order of summation and integration. ■

Remark 32. *Numerical evaluation of* (6.27) *indicates that* $\mathbb{E}\left[T_{0,FJ}^{(r,t)}(\gamma)\right]$ *is monotonically increasing in* γ *(see Fig. 6.8). This suggests that if one has limited service rate budget, it is optimal to allocate it entirely to the systematic node.*

6.6.2 Service Rate Allocation for $(t + 1)$-Split Model

In this model, a request is sent to the systematic node with probability $\alpha 0$ and is sent to its l-th repair groups with probability αl for $1 \leq l \leq t$. Notice that a request goes to



(a) $r = 2$

(b) $r = 3$

Figure 6.8: $\mathbb{E}\left[T_{0,\text{FJ}}^{(r,t)}(\gamma)\right]$ versus γ for various values of r.

exactly one of the repair groups or to the systematic node. In the following, we find the mean download time and the optimal allocation γ.

Proposition 10. *When the systematic node is allocated $1-\gamma$ fraction of the total service rate $\tilde{\mu}$ and the remaining rate $\gamma\tilde{\mu}$ is equally distributed among the repair group nodes $(0 \leq \gamma \leq 1)$, the mean download delay for an (r,t)-availability code using a $t+1$-split* *traffic model is*

$$\mathbb{E}\left[T^{(r,t)}_{t+1,FJ}(\gamma)\right] = \alpha0\frac{1}{(1-\gamma)\tilde{\mu}} + (1-\alpha0)\frac{rtH_r}{\gamma\tilde{\mu}}. \tag{6.29}$$

For a given routing probability $\alpha0$, the optimal delay is given as

$$\mathbb{E}\left[T^{(r,t)}_{t+1,FJ}(\gamma^*)\right] = \frac{1}{\tilde{\mu}}\left(\sqrt{\alpha0} + \sqrt{(1-\alpha0)rtH_r}\right)^2, \tag{6.30}$$

where $\gamma^* = \frac{\sqrt{(1-\alpha0)rtH_r}}{\sqrt{\alpha0}+\sqrt{(1-\alpha0)rtH_r}}$ is the optimal allocation.

Proof: Notice that with probability $\alpha0$, download time is the service time at the systematic node with service rate $(1-\gamma)\tilde{\mu}$. With the remaining probability $(1-\alpha0)$, download time is the service time at a repair group, which is equal to the service time for an (r,r) fork-join queue [141]. It is well-known that, for an (r,r) fork-join queue with each server having rate μ', the expected service time is $\frac{H_r}{\mu'}$ (see [122]), where $\mu' = \frac{\gamma\tilde{\mu}}{rt}$. This allows us to get (6.29).

We observe that (6.29) is convex, and thus, taking a derivative with respect to γ and equating to zero gives us γ^* and (6.30). ■

Remark 33. *Notice that* (6.30) *is an increasing function of t. This suggests that it is optimal to allocate all the γ fraction to a single repair group than spreading it among multiple repair groups.*

6.6.3 Service Rate Allocation for 2-Split Fork Join Model

In this model, systematic node gets $\alpha0$ fraction of requests, the rest of the requests are sent to all the t repair groups. At the repair groups, download time is $S^{(r,t)}$. We can find the mean download time under 2-split model as follows.

Proposition 11. *When the systematic node is allocated $1-\gamma$ fraction of the total service rate $\tilde{\mu}$ and the remaining service rate $\gamma\tilde{\mu}$ is equally distributed among the repair group* *nodes* $(0 \leq \gamma \leq 1)$*, the mean download delay for an* (r,t)*-availability code using a* 2*-split traffic model is*

$$\mathbb{E}\left[T_{2,FJ}^{(r,t)}(\gamma)\right] = \alpha 0 \frac{1}{(1-\gamma)\tilde{\mu}} + (1-\alpha 0)\frac{rtb^{(r,t)}}{\gamma\tilde{\mu}}, \tag{6.31}$$

where $b^{(r,t)} := \sum_{j=1}^{t} \binom{t}{j}(-1)^j H_{rj}$*.*

For a given routing probability $\alpha 0$*, the optimal delay is given as*

$$\mathbb{E}\left[T_{t+1,FJ}^{(r,t)}(\gamma^*)\right] = \frac{1}{\tilde{\mu}}\left(\sqrt{\alpha 0} + \sqrt{(1-\alpha 0)rtb^{(r,t)}}\right)^2, \tag{6.32}$$

where $\gamma^* = \frac{\sqrt{(1-\alpha 0)rtb^{(r,t)}}}{\sqrt{\alpha 0}+\sqrt{(1-\alpha 0)rtb^{(r,t)}}}$ *is the optimal allocation.*

Proof: With probability $\alpha 0$ a request will be served at the systematic node, while with probability $1-\alpha 0$, a request will be forked to all the t repair groups. Thus, we can write

$$\mathbb{E}\left[T_{2,\text{FJ}}^{(r,t)}\right] = \alpha 0 \mathbb{E}\left[S_i\right] + (1-\alpha 0)\mathbb{E}\left[S^{(r,t)}\right], \tag{6.33}$$

where S_i is the service time at the systematic node and $S^{(r,t)}$ is the service time by forking to all the t repair groups. Using (6.13) with service rate per node as $\mu' = \frac{\gamma\tilde{\mu}}{rt}$, and $\mathbb{E}\left[S_i\right] = \frac{1}{(1-\gamma)\tilde{\mu}}$, gives (6.31). Since (6.31) is convex, taking a derivative with respect to γ and equating to zero gives us γ^* and (6.32). ■

Remark 34. *Notice that* $b^{(r,t)} = \tilde{\mu}\mathbb{E}\left[S^{(r,t)}\right] = \mathbb{E}\left[X_{1:t}\right]$*, where* $X = S_{r:r}^{l}$ *is the service time at a repair group, which is the maximum of* r *i.i.d. exponential random variables with unit service rate. Recall that an exponential random variable is a log-concave random variable. (See Sec 6.2.) This allows us to show that* $S_{r:r}^{l}$ *is also log-concave. (Sec. 6.2, Claim 6.) Then, we can use a result from [126] that for a log-concave random variable* S*,* $t\mathbb{E}\left[S_{1:t}\right]$*, is a non-decreasing function of* t*, where* $S_{1:t}$ *is the minimum of* t *i.i.d. random variables each with PDF* $f_S(s)$*. Therefore, it is optimal to allocate* γ *fraction to only a* *single repair group.*

One needs to investigate whether it is desirable to allocate the service rate to more than one repair groups when the service time distribution is not exponential. In fact, it would be interesting to explore if conclusions can be drawn just by knowing the log-concavity/log-convexity property of the service time distributions, similar to [126].

6.7 Conclusion

We first studied the impact of the availability parameters, *i.e.*, the size and the number of repair groups, on the download latency by computing the mean download time of repair groups and the probability that repair groups perform slower download than the systematic nodes. Next, we characterized optimal service rate allocation across nodes when the total service capacity of the system is limited. We found that higher level of availability is not helpful in low arrival regime for exponential service time distribution, when the total service capacity of the system is limited.

Our results are likely to be found surprising since, in principle, higher availability should reduce delay by enabling multiple parallel reads. This indicates that we are still far from understanding the impact of availability on the download latency performance. On the other hand, the cost of availability on the minimum distance (and hence fault-tolerance) is well understood. Therefore, it would be very interesting to investigate when does availability reduce download latency by analyzing different scenarios such as when requests arrive at high rate, or when the service time distributions are different than exponential, or when decoding desired file from coded files takes non-negligible time.

7. CONCLUSIONS AND FUTURE DIRECTIONS

Large-scale cloud storage systems encounter a significant number of failure events due to various issues such as the use of commodity components, software glitches, power failures etc. In the face of the failures and the massive growth of data being stored and computed online, the task of designing reliable, efficient, available, and secure distributed storage systems opens up unique challenges and opportunities. In this thesis, we addressed some of these challenges and explored some new avenues by constructing novel, low-complexity coding schemes and analyzing fundamental limits of their performance metrics. However, there are a number of open problems, and we discuss potential future directions below.

7.1 Security in DSS

In Chapter 2, we presented a framework to jointly design outer and inner codes for achieving perfect security in DSS when the eavesdropper *Eve* can observe data stored on a limited number of nodes. There are several natural questions that one can ask. How to extend the framework to design secure regenerating codes when repair data is eavesdropped? How can one extend the framework for designing secure LRCs? How to use such secure LRCs to construct locally recoverable secret sharing schemes for multiple secrets? Note that we used longer storage codes to construct low field size secure codes. It is interesting to investigate whether one can utilize such a *code equivalence* to obtain lower bounds on field size for storage codes.

In Chapter 3, Section 3.4, we presented a universal outer code that can weakly secure any MSR code, when Eve can observe the data stored on any single storage node. The main feature of the proposed scheme is the lower alphabet-size requirement than the existing universal outer coding schemes. There are several natural questions that we are interested in investigating further. The proposed universal scheme assumes that Eve can observe any single storage node. On the other hand, the scheme based on maximum rank-distance (MRD) codes, proposed in [38], achieves weak security when Eve can control up to $k-1$ nodes. Notice that $k-1$ is the maximum possible number of nodes that can be eavesdropped. In many scenarios, it might be possible to estimate a bound on the number of nodes l that can be eavesdropped. For instance, suppose half of the nodes are in a trusted cloud system while the other half of the nodes are a part of untrusted cloud system. Then, this ensures $l \leq \frac{k}{2}$. When the parameter l is given, can we design universal outer codes at lower field size than the scheme of [38]? What is the optimum field size (fundamental lower bound on the field size) for a universal outer code achieving weak security? In [47], it is shown that the field size of the scheme using MRD codes is optimal for large range of parameters for any universal scheme achieving *perfect secrecy* in network coding. It is interesting to explore the optimal field size for the weak security model.

7.2 Locality in DSS

In Chapter 4, Section 4.4.2, we presented a family of locally repairable codes with *unequal all-symbol locality*. The construction is based on the maximum rank-distance (MRD) outer codes [44]. The use of the rank-metric codes greatly increases the required field size. In particular, for our constructions, the field size required is $\mathcal{O}\left(n^n\right)$, where n is the block-length of the code. Naturally, it is important to investigate whether it is possible to construct codes such that the field size is $\mathcal{O}\left(n\right)$. For codes with equal locality, Tamo and Barg [25] presented a landmark code construction such that the alphabet size is only slightly greater than the code length. Their construction can be considered as a generalization of Reed-Solomon (RS) codes. The idea is to obtain codewords as evaluations of specially constructed polynomials over a finite field at carefully chosen evaluation points. The construction reduces to RS codes if the locality r equals the dimension of the code. One immediate question is how to extend this technique to accommodate unequal all-symbol locality. It will also be interesting to generalize the notion of unequal locality to *unequal availability*, where different symbols have different number of repair groups, and explore the connections to Private Information Retrieval codes [144].

In Chapter 5, we studied rank-metric and subspace codes with locality constraints. We quantified the requirement of locality under the rank-metric by introducing a notion of rank-locality, and constructed codes with rank-locality that possess optimal rank-distance. We used the lifting construction to obtain subspace codes with locality from the rank-metric codes with locality. A natural question is to derive a Singleton-like upper bound on the subspace distance, and to explore how one can design optimal subspace codes. It is also interesting to explore further applications of such codes in diverse areas such as distributed storage systems operating over a noisy network, and for linear authentication systems as considered (without any locality constraint) in [145].

7.3 Availability in DSS

In Chapter 6, we analyzed the download delay performance of availability codes when nodes take random time to fetch their contents. In general, an intriguing open question is to investigate how the download delay metric interplays with the other metrics when the service times are random, and in what ways the structure of the underlying code affects the download delay. Since the coding and queuing schemes are inherently and intricately connected to each other, this opens up several research problems in the code design paradigm: What are some fundamental properties that a code should possess so that its download latency performance is good? Would overlapping repair groups be helpful in reducing download latency? If yes, how can one construct codes with overlapping repair groups? How does non-uniform availability, i.e., different symbols having different number of recovering sets, affect the download delay? How to construct codes with non-uniform availability, and how is their reliability performance (with respect to the minimum distance)?

REFERENCES

[1] *Cloud Storage Market by Solution, Service, Deployment Model, Organization Size, Vertical & Region - Global Forecast to 2021*. Markets and Markets, Sep 2016. [Online; "http://www.marketsandmarkets.com/Market-Reports/cloud-storage-market-902.html"].

[2] B. Kim, “Keeping up with: Cybersecurity, usability, and privacy,” *College and Research Libraries News*, vol. 77, pp. 442–451, Oct 2016.

[3] J. Dean, “Evolution and future directions of large-scale storage and computation systems at google,” in *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC ’10, 2010.

[4] D. Ford, F. Labelle, F. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, “Availability in globally distributed storage systems,” in *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2010.

[5] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The google file system,” *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 29–43, Oct. 2003.

[6] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The hadoop distributed file system,” in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–10, 2010.

[7] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, “Erasure coding in windows azure storage,” in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, USENIX ATC’12, 2012.

[8] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramachandran, "Network Coding for Distributed Storage Systems," *IEEE Transactions on Information Theory*, vol. 56, pp. 4539–4551, Sept. 2010.

[9] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Transactions on Information Theory*, vol. 58, pp. 6925–6934, Nov 2012.

[10] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 2000–2008, May 2007.

[11] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A Survey on Network Codes for Distributed Storage," *Proceedings of the IEEE*, vol. 99, pp. 476–489, Mar. 2011.

[12] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1243–1249, Sept 2009.

[13] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramachandran, "Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction," *IEEE Transactions on Information Theory*, vol. 57, pp. 5227–5239, Aug. 2011.

[14] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: Mds array codes with optimal rebuilding," *IEEE Transactions on Information Theory*, vol. 59, pp. 1597–1616, March 2013.

[15] B. Sasidharan, G. K. Agarwal, and P. V. Kumar, "A high-rate msr code with polynomial sub-packetization level," in *2015 IEEE International Symposium on Infor-*

mation Theory (ISIT), pp. 2051–2055, June 2015.

[16] M. Ye and A. Barg, "Explicit constructions of high-rate mds array codes with optimal repair bandwidth," *IEEE Transactions on Information Theory*, vol. 63, pp. 2001–2014, April 2017.

[17] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Transactions on Information Theory*, vol. 59, pp. 1597–1616, March 2013.

[18] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on*, pp. 79–86, July 2007.

[19] J. Han and L. Lastras-Montao, "Reliable memories with subline accesses," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pp. 2531–2535, June 2007.

[20] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *INFOCOM, 2011 Proceedings IEEE*, pp. 1215–1223, April 2011.

[21] D. Papailiopoulos and A. Dimakis, "Locally repairable codes," *IEEE Transactions on Information Theory*, vol. 60, pp. 5843–5855, Oct 2014.

[22] A. Rawat, O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 60, pp. 212–236, Jan 2014.

[23] A. Wang and Z. Zhang, "An integer programming-based bound for locally repairable codes," *IEEE Transactions on Information Theory*, vol. 61, pp. 5280–5294, Oct 2015.

[24] N. Silberstein, A. Rawat, O. Koyluoglu, and S. Vishwanath, "Optimal locally repairable codes via rank-metric codes," in *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 1819–1823, July 2013.

[25] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Transactions on Information Theory*, vol. 60, pp. 4661–4676, Aug 2014.

[26] P. Huang, E. Yaakobi, H. Uchikawa, and P. H. Siegel, "Binary linear locally repairable codes," *IEEE Transactions on Information Theory*, vol. 62, pp. 6268–6283, Nov 2016.

[27] A. Wang and Z. Zhang, "Repair locality with multiple erasure tolerance," *IEEE Transactions on Information Theory*, vol. 60, pp. 6979–6987, Nov 2014.

[28] A. Rawat, D. Papailiopoulos, A. Dimakis, and S. Vishwanath, "Locality and availability in distributed storage," in *2014 IEEE International Symposium on Information Theory (ISIT)*, pp. 681–685, June 2014.

[29] I. Tamo and A. Barg, "Bounds on locally recoverable codes with multiple recovering sets," in *2014 IEEE International Symposium on Information Theory (ISIT)*, pp. 691–695, June 2014.

[30] L. Pamies-Juarez, H. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pp. 892–896, July 2013.

[31] S. Pawar, S. E. Rouayheb, and K. Ramchandran, "On secure distributed data storage under repair dynamics," in *2010 IEEE International Symposium on Information Theory Proceedings (ISIT)*, (Austin), June 2010.

[32] S. Pawar, S. E. Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Transactions*

on Information Theory, vol. 57, pp. 6734–6753, Oct. 2011.

[33] N. B. Shah, K. V. Rashmi, and P. V. Kumar, “Information-theoretically secure regenerating codes for distributed storage,” in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pp. 1–5, Dec 2011.

[34] R. Zhu and W. Guo, “On the secure conditions for distributed storage systems,” in *2013 International Symposium on Network Coding (NetCod)*, pp. 1–6, June 2013.

[35] K. Bhattad and K. R. Narayanan, “Weakly secure network coding,” in *2005 International Symposium on Network Coding (NetCod)*, April 2005.

[36] P. F. Oliveira, L. Lima, T. T. V. Vinhoza, J. Barros, and M. Medard, “Coding for trusted storage in untrusted networks,” *IEEE Transactions on Information Forensics and Security*, vol. 7, pp. 1890–1899, Dec 2012.

[37] S. H. Dau, W. Song, and C. Yuen, “On block security of regenerating codes at the mbr point for distributed storage systems,” in *2014 IEEE International Symposium on Information Theory*, pp. 1967–1971, June 2014.

[38] J. Kurihara and Y. Miyake, “Securing distributed storage systems based on arbitrary regenerating codes,” *IEICE Communications Express*, vol. 2, no. 10, pp. 442–446, 2013.

[39] D. Silva and F. R. Kschischang, “Universal weakly secure network coding,” in *Proc. Information Theory Workshop*, (Greece), pp. 281–285, June 2009.

[40] S. Kadhe and A. Sprintson, “Security for minimum storage regenerating codes and locally repairable codes,” in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 1028–1032, June 2017.

[41] S. Goparaju, S. E. Rouayheb, R. Calderbank, and H. V. Poor, "Data secrecy in distributed storage systems under exact repair," in *International Symposium on Network Coding (NetCod)*, pp. 1–6, June 2013.

[42] A. Agarwal and A. Mazumdar, "Security in locally repairable storage," *IEEE Transactions on Information Theory*, vol. 62, pp. 6204–6217, Nov 2016.

[43] A. S. Rawat, "Secrecy capacity of minimum storage regenerating codes," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 1406–1410, June 2017.

[44] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Problems Inform. Transmission*, vol. 21, pp. 1–12, Jul 1985.

[45] L. H. Ozarow and A. D. Wyner, "The wire-tap channel II," *Bell Syst. Tech. Journ.*, vol. 63, pp. 2135–2157, 1984.

[46] P. Gopalan, C. Huang, B. Jenkins, and S. Yekhanin, "Explicit maximally recoverable codes with locality," *IEEE Transactions on Information Theory*, vol. 60, pp. 5245–5256, Sept 2014.

[47] D. Silva and F. R. Kschischang, "Universal secure network coding via rank-metric codes," *IEEE Transactions on Information Theory*, vol. 57, pp. 1124–1135, Feb. 2011.

[48] S. E. Rouayheb, E. Soljanin, and A. Sprintson, "Secure network coding for wiretap networks of type II," *IEEE Transactions on Information Theory*, vol. 58, pp. 1361–1371, March 2012.

[49] G. Hu, *Lower Bounds for Error-Correcting Codes with Local Recovery*. PhD thesis, Princeton University, 2017.

[50] S. Kadhe and A. Sprintson, "Weakly secure regenerating codes for distributed storage," in *2014 International Symposium on Network Coding (NetCod)*, pp. 1–6, June 2014.

[51] S. E. Rouayheb and E. Soljanin, "On wiretap networks II," in *Proc. Int. Symp. Information Theory*, (Nice, France), pp. 551–555, June 2007.

[52] J. Kurihara, T. Uyematsu, and R. Matsumoto, "Explicit construction of universal strongly secure network coding via mrd codes," in *2012 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 1483–1487, July 2012.

[53] M. Ye and A. Barg, "Explicit constructions of high-rate mds array codes with optimal repair bandwidth," *IEEE Transactions on Information Theory*, vol. 63, pp. 2001–2014, April 2017.

[54] B. Sasidharan, M. Vajha, and P. V. Kumar, "An explicit, coupled-layer construction of a high-rate msr code with low sub-packetization level, small field size and d<(n-1)," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 2048–2052, June 2017.

[55] S. Goparaju and R. Calderbank, "Binary cyclic codes that are locally repairable," in *2014 IEEE International Symposium on Information Theory (ISIT)*, pp. 676–680, June 2014.

[56] S. Kadhe and A. Sprintson, "Codes with unequal locality," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 435–439, July 2016.

[57] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A Survey on Network Codes for Distributed Storage," *Proceedings of the IEEE*, vol. 99, pp. 476–489, Mar. 2011.

[58] O. Khan, R. Burns, J. Park, and C. Huang, "In search of i/o-optimal recovery from disk failures," in *Proceedings of the 3rd USENIX Conference on Hot Topics in Stor-*

age and File Systems, HotStorage'11, (Berkeley), pp. 6–6, 2011.

[59] N. Prakash, G. Kamath, V. Lalitha, and P. Kumar, "Optimal linear codes with a local-error-correction property," in *2012 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 2776–2780, July 2012.

[60] G. Kamath, N. Prakash, V. Lalitha, and P. Kumar, "Codes with local regeneration and erasure correction," *IEEE Transactions on Information Theory*, vol. 60, pp. 4637–4660, Aug 2014.

[61] I. Tamo, D. Papailiopoulos, and A. Dimakis, "Optimal locally repairable codes and connections to matroid theory," in *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 1814–1818, July 2013.

[62] T. Ernvall, T. Westerbäck, and C. Hollanti, "Linear locally repairable codes with random matrices," *CoRR*, vol. abs/1408.0180, 2014. Online: "http://arxiv.org/abs/1408.0180".

[63] W. Song, S. H. Dau, C. Yuen, and T. Li, "Optimal locally repairable linear codes," *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 1019–1036, May 2014.

[64] M. Kuijper and D. Napp, "Erasure codes with simplex locality," *CoRR*, vol. abs/1403.2779, 2014. Online: http://arxiv.org/abs/1403.2779.

[65] A. Zeh and E. Yaakobi, "Optimal linear and cyclic locally repairable codes over small fields," in *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, April 2015.

[66] N. Silberstein and A. Zeh, "Optimal binary locally repairable codes via anticodes," in *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 1247–1251, June 2015.

[67] A. Rawat, D. Papailiopoulos, A. Dimakis, and S. Vishwanath, “Locality and availability in distributed storage,” in *2014 IEEE International Symposium on Information Theory (ISIT)*, pp. 681–685, June 2014.

[68] I. Tamo and A. Barg, “Bounds on locally recoverable codes with multiple recovering sets,” in *Information Theory (ISIT), 2014 IEEE International Symposium on*, pp. 691–695, June 2014.

[69] N. Prakash, V. Lalitha, and P. Kumar, “Codes with locality for two erasures,” in *Information Theory (ISIT), 2014 IEEE International Symposium on*, pp. 1962–1966, June 2014.

[70] B. Sasidharan, G. Agarwal, and P. Kumar, “Codes with hierarchical locality,” in *Information Theory (ISIT), 2015 IEEE International Symposium on*, pp. 1257–1261, June 2015.

[71] V. Cadambe and A. Mazumdar, “Bounds on the size of locally recoverable codes,” *IEEE Transactions on Information Theory*, vol. 61, pp. 5787–5794, Nov 2015.

[72] G. Joshi, Y. Liu, and E. Soljanin, “On the delay-storage trade-off in content download from coded distributed storage systems,” *Selected Areas in Communications, IEEE Journal on*, vol. 32, pp. 989–997, May 2014.

[73] N. B. Shah, K. Lee, and K. Ramchandran, “When do redundant requests reduce latency?,” *IEEE Transactions on Communications*, vol. 64, no. 2, pp. 715–722, 2016.

[74] A. Zeh and E. Yaakobi, “Bound and constructions of codes with multiple localities,” *arXiv*, vol. abs/1601.02763, 2016. Online: “http://arxiv.org/abs/1601.02763”.

[75] S. Kadhe, S. E. Rouayheb, I. Duursma, and A. Sprintson, “Rank-metric codes with local recoverability,” in *2016 54th Annual Allerton Conference on Communication,*

Control, and Computing (Allerton), pp. 1033–1040, Sept 2016.

[76] A. Rowstron and P. Druschel, “Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility,” *SIGOPS Oper. Syst. Rev.*, vol. 35, pp. 188–201, Oct. 2001.

[77] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The google file system,” *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 29–43, Oct. 2003.

[78] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, V. Sivakumar, L. Tang, and S. Kumar, “F4: Facebook’s warm BLOB storage system,” in *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI’14, pp. 383–398, 2014.

[79] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, “Xoring elephants: novel erasure codes for big data,” in *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB’13, pp. 325–336, 2013.

[80] P. Delsarte, “Bilinear forms over a finite field, with applications to coding theory,” *Journal of Combinatorial Theory, Series A*, vol. 25, no. 3, pp. 226 – 241, 1978.

[81] R. M. Roth, “Maximum-rank array codes and their application to crisscross error correction,” *IEEE Transactions on Information Theory*, vol. 37, pp. 328–336, Mar 1991.

[82] M. Blaum and J. Bruck, “Mds array codes for correcting a single criss-cross error,” *IEEE Transactions on Information Theory*, vol. 46, pp. 1068–1077, May 2000.

[83] P. Gill, N. Jain, and N. Nagappan, “Understanding network failures in data centers: Measurement, analysis, and implications,” in *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM ’11, pp. 350–361, 2011.

[84] S. Nath, H. Yu, P. B. Gibbons, and S. Seshan, "Subtleties in tolerating correlated failures in wide-area storage systems," in *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pp. 17–17, 2006.

[85] L. N. Bairavasundaram, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, G. R. Goodson, and B. Schroeder, "An analysis of data corruption in the storage stack," *Trans. Storage*, vol. 4, pp. 8:1–8:28, Nov. 2008.

[86] J. L. Hafner, V. Deenadhayalan, W. Belluomini, and K. Rao, "Undetected disk errors in raid arrays," *IBM J. Res. Dev.*, vol. 52, pp. 413–425, July 2008.

[87] M. Balakrishnan, A. Kadav, V. Prabhakaran, and D. Malkhi, "Differential raid: Rethinking raid for ssd reliability," *ACM Transactions on Storage*, vol. 6, pp. 4:1–4:22, July 2010.

[88] M. Blaum, J. L. Hafner, and S. Hetzler, "Partial-mds codes and their application to raid type of architectures," *IEEE Transactions on Information Theory*, vol. 59, pp. 4510–4519, July 2013.

[89] K. Greenan, D. D. E. Long, E. L. Miller, T. Schwarz, and A. Wildani, "Building flexible, fault-tolerant flash-based storage systems," in *Proceedings of the Fifth Workshop on Hot Topics in System Dependability (HotDep 2009)*, June 2009.

[90] K. W. Shum and Y. Hu, "Cooperative regenerating codes," *IEEE Transactions on Information Theory*, vol. 59, pp. 7229–7258, Nov 2013.

[91] A. S. Rawat, A. Mazumdar, and S. Vishwanath, "On cooperative local repair in distributed storage," in *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, pp. 1–5, March 2014.

[92] P. Gopalan, C. Huang, B. Jenkins, and S. Yekhanin, “Explicit maximally recoverable codes with locality,” *IEEE Transactions on Information Theory*, vol. 60, pp. 5245–5256, Sept 2014.

[93] P. Gopalan, G. Hu, S. Saraf, C. Wang, and S. Yekhanin, “Maximally recoverable codes for grid-like topologies,” *CoRR*, vol. abs/1605.05412, 2016. Online: “http://arxiv.org/abs/1605.05412”.

[94] J. S. Plank and M. Blaum, “Sector-disk (sd) erasure codes for mixed failure modes in raid systems,” *Trans. Storage*, vol. 10, pp. 4:1–4:17, Jan. 2014.

[95] M. Blaum, J. S. Plank, M. Schwartz, and E. Yaakobi, “Construction of partial mds and sector-disk codes with two global parity symbols,” *IEEE Transactions on Information Theory*, vol. 62, pp. 2673–2681, May 2016.

[96] V. Sridharan and D. Liberty, “A study of dram failures in the field,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC ’12, (Los Alamitos, CA, USA), pp. 76:1–76:11, IEEE Computer Society Press, 2012.

[97] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, “Feng shui of supercomputer memory positional effects in dram and sram faults,” in *2013 SC - International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–11, Nov 2013.

[98] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, “Memory errors in modern systems: The good, the bad, and the ugly,” in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS ’15, (New York, NY, USA), pp. 297–310, ACM, 2015.

[99] J. Colgrove, J. D. Davis, J. Hayes, E. L. Miller, C. Sandvig, R. Sears, A. Tamches, N. Vachharajani, and F. Wang, “Purity: Building fast, highly-available enterprise flash storage from commodity components,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’15, pp. 1683–1694, 2015.

[100] S.-W. Jun, M. Liu, S. Lee, J. Hicks, J. Ankcorn, M. King, S. Xu, and Arvind, “Bluedbm: An appliance for big data analytics,” *SIGARCH Comput. Archit. News*, vol. 43, pp. 1–13, June 2015.

[101] R. Koetter and F. R. Kschischang, “Coding for errors and erasures in random network coding,” *IEEE Transactions on Information Theory*, vol. 54, pp. 3579–3591, Aug 2008.

[102] D. Silva and F. R. Kschischang, “On metrics for error correction in network coding,” *IEEE Transactions on Information Theory*, vol. 55, pp. 5479–5490, Dec 2009.

[103] D. Silva, F. R. Kschischang, and R. Koetter, “A rank-metric approach to error control in random network coding,” *IEEE Transactions on Information Theory*, vol. 54, pp. 3951–3967, Sept 2008.

[104] M. Blaum, J. S. Plank, M. Schwartz, and E. Yaakobi, “Partial mds (pmds) and sector-disk (sd) codes that tolerate the erasure of two random sectors,” in *Information Theory (ISIT), 2014 IEEE International Symposium on*, pp. 1792–1796, June 2014.

[105] A. Khaleghi, D. Silva, and F. R. Kschischang, *Subspace Codes*, pp. 1–21. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[106] T. Etzion and N. Silberstein, “Error-correcting codes in projective spaces via rank-metric codes and ferrers diagrams,” *IEEE Transactions on Information Theory*,

vol. 55, pp. 2909–2919, July 2009.

[107] E. M. Gabidulin and M. Bossert, “Algebraic codes for network coding,” *Problems of Information Transmission*, vol. 45, pp. 343–356, Dec 2009.

[108] R. Ahlswede and H. Aydinian, “On error control codes for random network coding,” in *2009 Workshop on Network Coding, Theory, and Applications*, pp. 68–73, June 2009.

[109] M. Gadouleau and Z. Yan, “Constant-rank codes and their connection to constant-dimension codes,” *IEEE Transactions on Information Theory*, vol. 56, pp. 3207–3216, July 2010.

[110] F. Manganiello, E. Gorla, and J. Rosenthal, “Spread codes and spread decoding in network coding,” in *2008 IEEE International Symposium on Information Theory*, pp. 881–885, July 2008.

[111] T. Etzion and A. Vardy, “Error-correcting codes in projective space,” *IEEE Transactions on Information Theory*, vol. 57, pp. 1165–1173, Feb 2011.

[112] N. Silberstein and A. L. Trautmann, “Subspace codes based on graph matchings, ferrers diagrams, and pending blocks,” *IEEE Transactions on Information Theory*, vol. 61, pp. 3937–3953, July 2015.

[113] E. Ben-Sasson, T. Etzion, A. Gabizon, and N. Raviv, “Subspace polynomials and cyclic subspace codes,” *IEEE Transactions on Information Theory*, vol. 62, pp. 1157–1165, March 2016.

[114] N. Raviv and T. Etzion, “Distributed storage systems based on intersecting subspace codes,” in *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 1462–1466, June 2015.

[115] N. Silberstein, T. Etzion, and M. Schwartz, “Locality and availability of array codes constructed from subspaces,” in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 829–833, June 2017.

[116] F. J. MacWilliams and N. J. A. N. J. A. Sloane, *The theory of error correcting codes*. North-Holland mathematical library, Amsterdam, New York: North-Holland Pub. Co. New York, 1977.

[117] R. Lidl and H. Niederreiter, *Finite fields*. Encyclopedia of mathematics and its applications, New York: Cambridge University Press, 1997.

[118] M. Hall, Jr., *Combinatorial Theory (2Nd Ed.)*. New York, NY, USA: John Wiley & Sons, Inc., 1998.

[119] E. M. Gabidulin and N. I. Pilipchuk, “Error and erasure correcting algorithms for rank codes,” *Designs, Codes and Cryptography*, vol. 49, no. 1, pp. 105–122, 2008.

[120] S. Kadhe, E. Soljanin, and A. Sprintson, “Analyzing the download time of availability codes,” in *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 1467–1471, June 2015.

[121] S. Kadhe, E. Soljanin, and A. Sprintson, “When do the availability codes make the stored data more available?,” in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 956–963, Sept 2015.

[122] G. Joshi, Y. Liu, and E. Soljanin, “Coding for fast content download,” in *Allerton Conf.*, pp. 326–333, Oct. 2012.

[123] N. B. Shah, K. Lee, and K. Ramchandran, “The MDS queue: Analysing the latency performance of erasure codes,” in *2014 IEEE International Symposium on Information Theory (ISIT’14)*, pp. 861–865.

[124] G. Liang and U. C. Kozat, "Fast cloud: Pushing the envelope on delay performance of cloud storage with coding," *Networking, IEEE/ACM Transactions on*, vol. 22, no. 6, pp. 2012–2025, 2014.

[125] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia, "Reducing latency via redundant requests: Exact analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1, pp. 347–360, 2015.

[126] G. Joshi, E. Soljanin, and G. Wornell, "Efficient redundancy techniques for latency reduction in cloud systems," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 2, pp. 12:1–12:30, Apr. 2017.

[127] G. Joshi, E. Soljanin, and G. Wornell, "Queues with redundancy: Latency-cost analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 2, pp. 54–56, 2015.

[128] A. Kumar, R. Tandon, and T. Clancy, "On the latency and energy efficiency of distributed storage systems," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, 2015.

[129] B. Li, A. Ramamoorthy, and R. Srikant, "Mean-field-analysis of coding versus replication in cloud storage systems," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, April 2016.

[130] I. Tamo and A. Barg, "Bounds on locally recoverable codes with multiple recovering sets," in *Information Theory (ISIT), 2014 IEEE International Symposium on*, pp. 691–695, June 2014.

[131] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *Proceedings of the 2012 USENIX*

Conference on Annual Technical Conference, USENIX ATC'12, 2012.

[132] A. Wang and Z. Zhang, "Repair locality with multiple erasure tolerance," *Information Theory, IEEE Transactions on*, vol. 60, pp. 6979–6987, Nov 2014.

[133] Y. Sun, Z. Zheng, C. E. Koksal, K. Kim, and N. B. Shroff, "Provably delay efficient data retrieving in storage clouds," in *IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2015.

[134] Y. Xiang, T. Lan, V. Aggarwal, and Y. F. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 2443–2457, Aug 2016.

[135] K. Lee, R. Pedarsani, and K. Ramchandran, "On scheduling redundant requests with cancellation overheads," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 99–106, Sept 2015.

[136] K. Gardner, M. Harchol-Balter, and A. Scheller-Wolf, "A better model for job redundancy: Decoupling server slowdown and job size," in *IEEE Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2016)*, 2016.

[137] M. Bagnoli and T. Bergstrom, "Log-concave probability and its applications.," *Economic Theory*, vol. 26, no. 2, pp. 445–469, 2005.

[138] K. Lee, N. B. Shah, L. Huang, and K. Ramchandran, "The mds queue: Analysing the latency performance of erasure codes," *IEEE Transactions on Information Theory*, vol. 63, pp. 2822–2842, May 2017.

[139] S. Boucheron, G. Lugosi, P. Massart, and M. Ledoux, *Concentration inequalities: a non-asymptotic theory of independence*. Oxford: Oxford university press, 2013.

[140] S. M. Ross, *Introduction to Probability Models, Ninth Edition*. Academic Press, Inc., 2006.

[141] R. Nelson and A. Tantawi, “Approximate analysis of fork/join synchronization in parallel queues,” *Computers, IEEE Transactions on*, vol. 37, pp. 739–743, Jun 1988.

[142] E. Varki, A. Merchant, and H. Chen, “The m/m/1 fork-join queue with variable sub-tasks,” *unpublished, available online: "http://www.cs.unh.edu/ varki/publication/2002-nov-open.pdf"*.

[143] H. Tijms, *A First Course in Stochastic Models*. Wiley, 2003.

[144] A. Fazeli, A. Vardy, and E. Yaakobi, “Codes for distributed pir with low storage overhead,” in *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 2852–2856, June 2015.

[145] H. Wang, C. Xing, and R. Safavi-Naini, “Linear authentication codes: bounds and constructions,” *IEEE Transactions on Information Theory*, vol. 49, pp. 866–872, April 2003.