Imperial College London

Department of Electrical and Electronic Engineering

Coding-Theoretic Approaches to Distributed Caching, Storage and Computing

Nitish Mital

February 2021

Supervised by Dr. Cong Ling
Co-supervised by Dr. Deniz Gündüz

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London
and the Diploma of Imperial College London

Abstract

The next-generation communication networks aim to bring data storage and computation closer to the end users and Internet of things (IoT) endpoints by exploiting edge servers and/or memory available at edge devices. This greatly improves the efficiency and economics of network applications. Traditionally, coding techniques have been applied to communication problems with terrific gains in performance. In this thesis, we apply novel coding techniques to edge storage and computation problems, and show similar significant gains in terms of storage costs, communication costs, computation costs, and privacy.

In the first part of the thesis, we study a wireless distributed network of small base stations (SBS) which serve cache-enabled users. We propose a novel scheme of coded caching in a multi-server system with a random connectivity pattern, and study the fundamental limits of the trade-off between the storage capacity at the servers, the cache capacity at the users, and the delivery latency.

We then study the trade-off between the storage capacity and bandwidth requirement for the repair of failed storage nodes/SBSs in a distributed wireless content caching system. We derive the fundamental limits of this trade-off using techniques from information theory, and propose a storage and repair framework to achieve the optimal trade-off, while satisfying the practical constraints of low subpacketization, low overheads, and low computational complexity.

We then address problems related to distributed edge computing systems. In particular, we address computational privacy problems, where a user offloads the computation of a function on a set of matrices to powerful distributed servers, while preserving the privacy of the data and the computation result from T colluding servers as well as the user, and also achieving significantly smaller communication costs than existing secure multiparty computation techniques.

Declaration of Originality

I hereby certify that this thesis is the result of my own work under the guidance of my Ph.D. advisor, Dr. Cong Ling, and co-supervisor Dr. Deniz Gündüz. Any ideas or quotations from the work of other people are appropriately referenced.

Imperial College London
London, United Kingdom
10 February 2021

Nitish Mital

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor, Dr. Cong Ling, and my co-supervisor Dr. Deniz Gündüz, for giving me constant guidance and precious advice during my doctorate. Thanks to their vast knowledge and passion for research, they have been an incredible source of motivation and inspiration for me, both from a professional and a human point of view. I have to thank them for passing their passion for research on to me. Moreover, I would like to acknowledge the financial support provided by the European Union's H2020 research and innovation programme under the Marie Sklodowska-Curie Action SCAVENGE (grant agreement no. 675891), and the European Research Council (ERC) Starting Grant BEACON (grant agreement no. 677854) for funding my Ph.D. studies, and for enabling me to study in such an excellent research environment. I would also like to thank the examiners for the interesting and useful discussion we had during the viva examination, which greatly improved the quality of this work.

I have to thank all the people in the Scavenge team and the coordinators of the programme. I thank my friends and colleagues from the communications and signal processing group, as well from the information processing and communications lab, for their friendship, including uncountable coffee breaks, dinners, and interesting conversations. I also acknowledge the help of the research administrators of the two groups in managing my research smoothly. Thank you also to all of my new and old friends, with whom I have shared the best experiences of these years.

I also want to express my deepest gratitude to my Guru, Paramahamsa Sri Nithyananda Paramashivam, my parents and my brother, Atharv, for their constant support throughout the ups and downs.

London, United Kingdom
10 February 2021

Nitish

śānti mantraḥ

ओं स॒ह नाऽव॑वतु ।
स॒ह नौ॑ भुनक्तु ।
स॒ह वी॒र्यं॑ करवावहै ।
ते॒ज॒स्विना॒वधी॑तमस्तु मा
वि॑द्विषा॒वहै॑ ॥
ओं शान्ति॒: शान्ति॒: शान्ति॑: ॥

oṁ saha nā́vavatu |
saha nau᳡ bhunaktu |
saha vīryā́ṁ karavāvahai |
tejasvināvadhī́tamastu mā
vi᳡dviṣāvahai᳡ ॥
oṁ śāntiḥ śāntiḥ śāntiḥ ॥

Contents

List of Figures **11**

List of Acronyms **13**

Notations **14**

1 Introduction **15**
1.1 Outline and Contributions 18
1.1.1 Efficient content delivery using coded caching 18
1.1.2 Regenerating codes for distributed storage 20
1.1.3 Secure distributed matrix computation 24
1.2 Publications 29

2 Coded Caching in a Multi-Server System **31**
2.1 Introduction 31
2.1.1 Choice of connectivity parameter ρ 33
2.1.2 Centralized coded caching with single server 33
2.1.3 Coded caching with multiple servers 34
2.2 Problem Setting 35
2.2.1 Formal problem statement 37
2.3 Coded Distributed Storage and Caching Scheme 39
2.3.1 Server storage placement 40
2.3.2 User cache placement 41
2.3.3 Delivery phase 41
2.4 Successive SBS Transmissions 42
2.4.1 Redundancy in server storage capacity 45
2.4.2 Performance analysis 47
2.5 Lower Bound 49
2.5.1 Redundancy in server storage 53
2.6 Parallel SBS Transmissions 55
2.6.1 Redundant server storage capacity 55
2.7 Results and Discussions 56
2.8 Conclusions and Future Work 60

3 Regenerating Codes for Distributed Storage **61**
3.1 Introduction 61
3.1.1 Regenerating codes 62
3.1.2 Exact vs functional repair 64
3.1.3 Related work 64
3.1.4 Our contributions 67
3.2 System Model 68

3.2.1 Information flow graph 69
3.3 Storage-Bandwidth Trade-off for Partial Repair 71
3.4 Code Construction 76
3.4.1 Subspace view 77
3.4.2 Linearized polynomials 78
3.4.3 General code construction for any point on the trade-off curve with $\rho = 0$ (full-node repair) 79
3.4.3.1 Conditions for an optimal scheme 80
3.4.3.2 Reconstruction 80
3.4.3.3 Proposed code construction 82
3.4.3.4 Repair scheme 83
3.4.4 MSR point with $\rho = 0$ 85
3.4.5 MBR point with $\rho = 0$ 85
3.4.6 Code construction for any point on the trade-off curve with $\rho > 0$ (partial repair) 86
3.4.6.1 Conditions for an optimal scheme 86
3.4.6.2 Reconstruction 87
3.4.6.3 Proposed code construction 87
3.4.6.4 Repair scheme 88
3.5 Results and Discussion 88
3.5.1 Optimal trade-off curve 88
3.5.2 Performance of the code construction 89
3.5.2.1 Achievable storage-repair bandwidth pairs 89
3.5.2.2 Subpacketization 90
3.5.2.3 Input/Output cost 91
3.5.2.4 Computational complexity 91
3.6 Conclusions 92

4 Secure Distributed Matrix Computations 93
4.1 Introduction 93
4.1.1 Related Work 95
4.1.2 Main contributions 99
4.1.3 Data representation using finite fields 99
4.1.3.1 Finite field 100
4.1.3.2 Encoding integers 100
4.1.3.3 Encoding fractional numbers 100
4.1.3.4 Round-off errors 101
4.1.3.5 Computations on real data using finite fields 101
4.2 System Model 102
4.3 SDMM for $\Gamma = 2$ 105
4.3.0.1 Sharing phase 105
4.3.0.2 Computation phase 106
4.3.0.3 Reconstruction phase 107
4.3.1 Special case: Computation with own data 108
4.3.2 Proof of security against the servers 110

4.3.3 Security against the user . . . 110
4.3.4 Other performance metrics . . . 111
4.3.4.1 Encoding complexity . . . 111
4.3.4.2 Download cost . . . 111
4.3.4.3 Decoding complexity . . . 112
4.3.5 Comparison with other schemes . . . 113
4.4 Straggler Mitigation . . . 114
4.4.1 Sharing phase . . . 116
4.4.2 Computation phase . . . 116
4.4.3 Reconstruction phase . . . 116
4.4.4 Special case: computation with own data . . . 117
4.4.5 Performance Analysis: . . . 117
4.5 Secure Distributed Multiplication of Multiple Matrices . . . 118
4.5.1 Sharing phase . . . 119
4.5.2 Computation phase . . . 119
4.5.3 Communication phase . . . 119
4.5.4 Reconstruction phase . . . 120
4.5.5 Performance analysis . . . 120
4.6 Secure Matrix Algebra . . . 121
4.6.1 Matrix addition and multiplication by a scalar . . . 121
4.6.2 Changing from (N, K_1, T_1) left-shares to (N, K_2, T_2) right-shares 121
4.6.3 Transpose of a matrix . . . 123
4.6.4 Exponentiation . . . 124
4.6.5 Solving the linear system $\mathbf{AX} = \mathbf{B}$ with secure Gaussian elimination . . . 125
4.6.6 Secure matrix inversion . . . 125
4.6.7 Iterative matrix inversion . . . 126
4.6.8 Computation of arbitrary polynomials: . . . 127
4.6.9 Secure learning from local datasets . . . 128
4.6.10 Achieving the optimal upload and download cost for SDMM . . 128
4.7 SDMC in Integer Rings . . . 129
4.7.1 Finding the principal N^{th} roots of unity . . . 130
4.7.2 Examples . . . 132
4.8 SDMM in Integer Rings . . . 132
4.8.1 N not coprime to n . . . 133
4.8.2 Comparison with existing schemes . . . 133
4.9 Discussion and Conclusions . . . 134

5 Conclusions **136**
5.0.1 Research Challenges . . . 138

Bibliography **141**

List of Figures

1.1 Architecture of an edge network (NoMore201 / CC BY-SA (https://creativecommons.org/licenses/bu-sa/4.0)) . 16
1.2 A typical edge server, not much bigger than a notebook, with 256GB RAM and 16TB storage capacity. It can be hung on a wall or stored in a rack. (Image: Lenovo) 16

2.1 Maddah-Ali-Niesen (MAN) scheme for 3 users requesting files $\{W_1, W_2, W_3\}$ respectively, each of size F bits, and having cache capacity of $M_U = F$ bits. Each file is divided into 3 segments of size $\frac{F}{3}$ each. Each user caches one segment of each file, thus satisfying the cache constraint. All the requests are satisfied with 3 transmissions, i.e., $3 \times \frac{F}{3} = F$ bits transmitted on the channel. 33
2.2 Examples of different network topologies for $P = 3$ and $K = 4$ with $\rho = 2$. 36
2.3 Segmentation, MDS coding and placement of files. 40
2.4 An example 7×5 incidence matrix ($P = 7, K = 5$) with $\rho = 4$. 46
2.5 The ordering of servers to count the minimum cover. The dashed line indicates the point at which enough servers have been counted to deliver $\hat{\alpha}$ coded subsegments to all users in $\mathcal{H}$. 47
2.6 Toy example with $P = 2, K = N = M_S = 3, M_U = 1$ 54
2.7 Average normalized delivery latency vs. user cache capacity M_U, for $P = 7, N = K = 5, \rho = 4$, and for server storage capacities of $M_S = \frac{5}{4}, \frac{5}{3}, \frac{5}{2}, 5$. . . . 56
2.8 Average normalized delivery latency vs. server storage capacity M_S, for $P = 7, N = K = 5, M_U = 1$ for successive SBS transmissions. 57
2.9 Comparing the simulation with the theoretical, Average normalized delivery latency vs. server storage capacity, for $P = 21, N = K = 5, M_U = 1$ for successive SBS transmissions. 58
2.10 Average normalized delivery latency for parallel transmissions vs. user cache capacity M_U, for $P = 7, N = K = 5, \rho = 4$, and for server storage capacities of $M_S = \frac{5}{4}, \frac{5}{3}, \frac{5}{2}, 5$. 59
2.11 Average normalized delivery latency vs. server storage capacity M_S, for $P = 7, N = K = 5, M_U = 1$ for parallel transmissions. 59

3.1 A distributed SBS cache scenario. 62
3.2 Information flow graph $\mathcal{G}$ with $n = 4, k = 2, r = 2$, one repair round and cuts χ_1, χ_2 and χ_3. 72
3.3 Trade-off curve between the repair bandwidth and storage, $M = 1, k = 8$ and $d = 10$ helper nodes. For single node failure $r = 1$ and for multiple node failures $r = 2$. 89

4.1 System model for Γ matrices with N servers, any T of which may collude (shown as shaded servers). 103

4.2 Comparison of the upload cost of SDMM of two matrices for different number of colluding servers out of a total of $N = 20$ servers. Only the values of $T \leq 9$ are considered since the proposed scheme is defined only for $T < N/2$. The maximum value of K is calculated for each T. 114
4.3 Conversion from left-share to right-share (right-share to left-share conversion can be done similarly). 122

List of Acronyms

CDN	Content Delivery Network
5G	Fifth Generation
MDS	Maximum Distance Separable
SBS	Small Base Station
AWS	Amazon Web Services
HPC	High Performance Computing
ML	Machine Learning
MAN	Maddah Ali & Niesen
SDMC	Secure Distributed Matrix Commputation
SDMM	Secure Distributed Matrix Multiplication
MPC	Multi Party Computation
MSR	Minimum Storage Regeneration
MBR	Minimum Bandwidth Regeneration
RS	Reed-Solomon
BGW	Ben-Or, Goldwasser and Widgerson
IoT	Internet of Things
NIST	National Institute of Standards and Technology
3GPP	3^{rd} Generation Partnership Project
AR	Augmented Reality
VR	Virtual Reality
LTE	Long Term Evolution
RAN	Radio Access Network
NFV	Network Function Virtualization
SDN	Software Defined Networking

Notations

Throughout this dissertation we will use the following notation. For two integers $i < j$, we denote the set $\{i, i+1, \ldots, j\}$ by $[i : j]$, while the set $[1 : j]$ is denoted by $[j]$. Sets are denoted with the calligraphic font, and $|\mathcal{A}|$ denotes the cardinality of set $\mathcal{A}$. For $\mathcal{A} = \{a_1, a_2, \ldots, a_p\}$, we define $X_{\mathcal{A}} \triangleq (X_{a_1}, \ldots, X_{a_p})$. We denote a vector with a bold lower case letter $\mathbf{v}$, while we denote a matrix with a bold upper case letter $\mathbf{V}$. $\mathbb{1}_E$ denotes the indicator function of the event E, i.e., its value is 1 when the event E happens. $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

Chapter 1

Introduction

The next-generation of cellular communication networks, the so called fifth-generation or 5G mobile network, promises a tremendous increase in data speeds, ultra-low latency, more reliability, massive network capacity, increased availability, and a more uniform experience to users. A key enabling architectural innovation in 5G networks is bringing data storage and computation relatively closer to the end users and/or Internet of Things (IoT) endpoints. This greatly improves the efficiency and the economics of applications. Content delivery networks (CDN) increase network performance and uptime by caching frequently-accessed content on local servers that are positioned at the Internet's edge. CDN edge servers provide content such as images, JavaScript, HTML, and downloadable content. These servers are deployed at points of presence (PoPs) and edge locations across a CDN. Edge servers play an important role in implementing this architecture by providing storage and compute resources close to the end user, thus lowering the load on the backhaul and the core network. The wireless CDN has also been referred to as *femtocaching* [1], where content is cached at small base stations. CDNs are a vital backbone for much of the data that traverses the Internet. In fact, the majority of web traffic today is served through CDNs. As a result, CDN servers help reduce the workload on origin servers and reduce latency for users.

Due to the popularity of video traffic and the large size of the involved files, video content delivery is currently at the focal point of caching research, where caching at the user side, also called buffering, is leveraged to improve video delivery and avoid video stalls. This is just one example out of a multitude of applications and design policies proposed for caching. The Akamai intelligent edge platform, which owns 20-30% of the world's CDN traffic, the Google global cache (GCC), which comprises the caches installed at Internet service provider (ISP) premises, the Netflix open connect, Facebook photo CDN, Amazon AWS, and Cadami are some industrial implementations

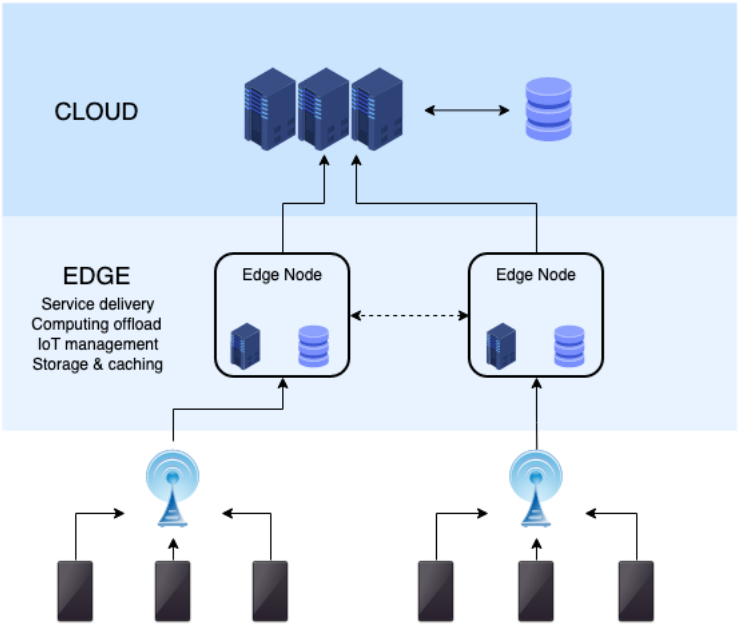


FIGURE 1.1: Architecture of an edge network
(NoMore201 / CC BY-SA (https://creativecommons.org/licenses/bu-sa/4.0))



FIGURE 1.2: A typical edge server, not much bigger than a notebook, with 256GB RAM and 16TB storage capacity. It can be hung on a wall or stored in a rack. (Image: Lenovo)

of large scale CDNs. We refer the reader to [2] for a comprehensive review of the role of caching CDNs in future wireless communication networks, and the state-of-art implementations.

In addition to CDN edge servers caching data, data may also be cached in the memories of user devices. Since the storage capabilities of end devices have increased manifold, they are capable of caching more data. The caches of these devices are populated in a pro-active manner in the *placement phase* during times of low network traffic, which reduces the amount of data that is required to be communicated when

the user requests a file during peak network traffic periods. This significantly reduces the network congestion and the end-to-end delays.

Edge computing servers provide computing resources on the network edge. Like edge caching servers, edge computing servers are strategically deployed to reduce latency by reducing the geographical distance between the producers and consumers of data. However, as opposed to serving static web content, edge computing servers provide functionality such as data processing for IoT and 5G networks. Such edge computing services provided by Microsoft Azure, Google Cloud, Amazon AWS, and other similar platforms are being used by research institutions for scientific research, where computations on massive datasets like neural network training or signal processing are offloaded to a large number of distributed edge computing servers, which perform the computations in parallel, thus speeding up the computation time significantly.

Conventional storage and computing algorithms employ techniques from coding theory for greater efficiency. Codes are a system of rules and sequences to convert raw information into another form that is either shortened, redundant or secret, so that it can be communicated on a communication channel or stored in a storage medium. Coding theory is the study of codes and their fitness for specific applications. Coding theory has been ubiquitously applied in communications for data compression (source coding), error detection and correction (channel coding), data transmission (modulation/line coding), data storage, and cryptography, as well as finding inter-disciplinary applications in biology, physics, etc. Novel codes designed for specific 5G use cases may provide the promised performance of 5G networks.

Therefore, coding theory naturally provides tools for addressing many challenges occurring in 5G edge networks. 5G networks are projected to see a tremendous increase in the number and density of these edge servers. The edge storage and computing market is forecasted to grow significantly in the coming decade, with billions of edge servers deployed at a number of locations to provide ubiquitous distributed storage and computing services. It is not surprising that such large-scale systems inevitability face maintenance issues, and make efficient network resource sharing, user privacy and security compulsory. The information theory and communications research community has recently invested considerable interest and efforts in solving these challenges.

1.1 Outline and Contributions

This dissertation illustrates the use of coding-theoretic techniques in communication, storage and computing at the edge network. In particular, we focus on content delivery from the edge servers to cache-equipped users, communication between storage servers for maintenance and repair, and communication between the users and edge servers for securely and robustly offloading computations to distributed servers. Compared to uncoded techniques, coded techniques are often able to achieve optimal trade-offs between communication cost, computing cost, storage capacity, and security/privacy. In the following sections, we provide an outline of the various problems considered in this dissertation, report the existing literature related to those problems, and mention our contributions. All the proposed solutions to the different problems employ a range of similar techniques of partitioning data and then coding over the partitions with novel codes.

1.1.1 Efficient content delivery using coded caching

The caching problem in its entirety includes decisions about server placement, content placement, content discovery and content delivery. The objective of the server and content placement problem is to ensure the maximum availability of the most frequently accessed content in the nearest caches to the user. The users may also be equipped with caches, where the most frequently used, the most recently used, or the most popular content may be cached locally by the users to provide instant access and high throughput. Coded caching and distributed storage have received significant attention in recent years to exploit the available memory space and processing power of individual network nodes to increase the throughput and efficiency of data availability. With proactive caching, part of the data can be pushed to nodes' local cache memories during off-peak hours, called the *placement phase*, to reduce the burden on the network during peak traffic periods, called the *delivery phase* [3–14]. Coded caching refers to careful coded or uncoded placement of cache contents in the placement phase, and coded delivery on a multicast channel during the delivery phase. It was shown by Maddah Ali et. al. [3] that coded delivery on multicast links to cache-equipped

users helps to leverage a *global caching gain* which provides significant improvement in transmission efficiency over the *local caching gain* provided by uncoded caching. Global caching gain refers to the fact that by using coded caching, the gain is proportional to the sum of the cache sizes of all the participating users, while the gain provided by uncoded caching is proportional to the cache size of one user.

A different type of coded caching also improves the delivery performance in the so-called "femtocaching" scenario [6], where multiple cache-equipped servers, modeled as small-cell base stations (SBSs), collaboratively deliver contents to users. Coding for distributed storage systems has been extensively studied in the literature (see, for example, [15]), and in the femtocaching scenario, ideal maximum distance separable (MDS) codes allow users to recover contents by collecting parity bits from only a subset of SBSs they connect to [6], which also increases the chance of users locating content in the nearby caches. Coded caching in multi-server settings was studied first in [12], where the authors study coded caching schemes presented in [3] and [11] when parity servers are available. The authors consider special scenarios with one and two parity servers. They propose a scheme that stripes the files into blocks, and codes them across the servers with a systematic MDS code, and they also propose a scheme for the scenario in which files are stored as whole units in the servers, without striping. In [13], the authors also consider multiple servers, each having access to all the files in the library, serving the users through an intermediate network of relays. They consider the so-called *linear network* model, in which the network topology is fixed but unknown at the relay nodes. The authors study the delivery latency considering parallel transmissions from the servers, and show that there is a gain from using multiple servers when the relay nodes employ simple random linear network coding which makes the network transfer matrix $\mathbf{H}$ rich.

Another line of related works study caching in combination networks [14], [16], which consider a single server serving cache-equipped users through multiple relay nodes. The server is connected to these relays through unicast links, which in turn serve a distinct subset of a fixed number of users through unicast links. A combination network with cache-enabled relay nodes is considered in [16].

In Chapter 2, we combine distributed storage at the SBSs, similar to the "femto-caching" framework [6], with cache storage at the users, and consider coded delivery over error-free shared broadcast links [4]. We consider a library of N files stored across P SBSs, each equipped with a limited-capacity storage space. Unlike the existing literature, we consider a boolean random connectivity model [7]: during the delivery phase, each user connects only to a random subset of ρ SBSs, where $\rho \leq P$. This may be due to the density of distribution of SBSs, physical variations in the channel, or due to resource constraints. The boolean random connectivity model is a good first-order approximation of a random isotropic wireless network. Most importantly, the connections that form the network topology are not known in advance during the placement phase; therefore, the cache placement cannot be designed for a particular network topology. Storing the files across multiple SBSs, and allowing users to connect randomly to a subset of them results in a loss in multicasting opportunities for the servers, indicating a trade-off between the coded caching gain and the flexibility provided by distributed storage across the servers, which, to the best of our knowledge, has not been studied before. More importantly, we show that increasing the storage capacity at the servers helps in mitigating the effect of the lost multicasting opportunities, thus approaching the optimal performance of a centralized coded caching scheme. We prove, using the fact that the coded caching problem can be mapped to a set of index coding problems on a side information graph, where the cache contents are viewed as the side information, that the proposed coded delivery scheme is optimal when the cache contents are uncoded.

1.1.2 Regenerating codes for distributed storage

The SBS caches considered in Chapter 2 can be modeled as a distributed edge caching system that stores a library of popular files. The literature on distributed caching systems focuses mostly on the code design or the resource allocation for efficient storage of popular contents, assuming reliable cache nodes. Maximum distance separable (MDS) codes are typically used for distributed caching of contents at multiple access points [1, 17, 18]. MDS codes provide flexibility for storage so that users with different connectivity or mobility patterns can download a file from only a subset

of the access points. In particular, an (n, k) MDS code encodes a file of size M bits by splitting it into k equal-size packets and encoding them into n packets which are stored at n cache nodes. Each data packet consists of a set of symbols in a finite field $\mathbb{F}_q$. The original file can be reconstructed by accessing any k out of n packets from k distinct access points. This property is called the *data reconstruction property.*

However, storage devices are often unreliable and prone to failures; thus, efficient repair techniques that guarantee continuous data availability are essential for a successful implementation of distributed caching and content delivery techniques in practice. When some nodes partially or fully fail, or when content has to be moved to new cache nodes, their cache contents must be regenerated, either in the same failed node, or in the new cache nodes where the content has to be moved, to be able to continue serving users with the same amount of redundancy. An important objective of edge caching in wireless networks is to reduce the backhaul link loads; therefore, we consider *cache recovery at the edge*; that is, rather than updating the failed cache contents from a central server through backhaul links, the failed cache contents are regenerated with the help of surviving cache nodes. This is called the *data regeneration property* of a code. The total amount of data transferred from the surviving nodes to repair the failed nodes is called the *repair bandwidth. Regenerating codes* are codes that satisfy the data reconstruction property and the data regeneration property. They treat the data stored on each node as a vector of S data packets. In particular, a file of size M bits is split into P data packets consisting of symbols in the finite field $\mathbb{F}_q$. A total of S coded packets are stored on each node having a storage capacity of α bits. Similarly to existing literature [19–23], we refer to S as subpacketization.

Dimakis *et al.* showed in [15] that there is a fundamental trade-off between the storage capacity α and the repair bandwidth γ by mapping the repair problem in a distributed storage system to a multicasting network coding problem [24] over an information flow graph [25]. The trade-off is shown to be characterized by the cut-set bound for linear network coding, and can be achieved with high probability using random linear coding [24]. Similarly to what is observed in Chapter 2, one observes that the repair bandwidth decreases very fast with increasing server storage capacity. The analysis in [15] focuses on a single node repair, that is, losing one node triggers the

repair process. Subsequent works analyse this fundamental trade-off for multiple node repair, where the repair process is triggered only after r nodes fail. The two extreme points of this trade-off, called the minimum-storage regenerating (MSR) point and the minimum-bandwidth regenerating (MBR) point, have been studied in detail in literature.

Another terminology frequently found in the literature is that of *exact repair*, where the content of the repaired nodes is exactly the same as that of the failed nodes, and *functional repair*, where the content of the repaired nodes may be different from that of the failed nodes, while still satisfying the data reconstruction and data regeneration property.

Initially, most literature considered the exact repair of a single node at a time [19, 20, 23, 26–28]. The product-matrix framework proposed in [28] constructs optimal repair MSR and MBR codes, where the MBR code meets the cutset bound for all admissible parameters, but the MSR code meets the cutset bound only for parameters with $d \geq 2k-2$. Other works include Zig-zag codes [19], which require exponential subpacketization for achieving the MSR point, and Reed-Solomon code based constructions [20–23], which also require exponential subpacketization for meeting the cutset bound at the MSR point.

In [29], it was observed that multiple node repair is more efficient in terms of the repair bandwidth per node. Therefore, subsequent works [30–36], introduce cooperative regenerating codes, which repair multiple failures cooperatively by allowing each of the r nodes being repaired to collect data from any $n-r$ surviving nodes, and then to cooperate with the other $r-1$ nodes being repaired. Minimum storage cooperative regenerating (MSCR) codes have been proposed for a limited set of parameters in [32, 33], while, in [34], MSCR codes have been proposed for all parameters, but require exponential subpacketization. In [36], the authors gave an explicit construction of minimum bandwidth cooperative regenerating (MBCR) codes for all parameters n, k, d, r, employing only a linear subpacketization.

Instead, similarly to [37, 38], we consider broadcast repair; that is, transmissions from each node are received in an error-free manner by all the other nodes, and, we

consider the *partial repair* problem [37], in which a node loses only a part of its contents, and the remainder of the contents may be used along with the transmissions from the surviving nodes to repair the content, thus further reducing the repair bandwidth. The remaining contents in a partially failed node may be viewed as side information, similarly to the caching problem studied in Chapter 2, that can be used to repair the erased content. In [37], the authors derive a lower bound on the number of packet transmissions at the MSR point for error-free broadcast partial repair, and provide an explicit code construction for a special case. However, the information flow graph construction in [37] does not capture the relation between storage capacity and the repair bandwidth, thus focusing only on one of the extreme points on the storage-repair bandwidth trade-off curve, corresponding to the MSR point. In this dissertation, we obtain the entire optimal trade-off curve. Other related works include [39–42], which study similar or equivalent repair problems.

While the exact repair problem has been studied extensively, the functional repair problem has not seen much progress. Although random linear coding is asymptotically optimal for functional repair [15, 43], it is not practically feasible due to the large overheads and high computational complexity of the Gaussian elimination-based decoding [44]. It was shown in [45] that a large portion of the functional repair trade-off curve cannot be achieved under exact repair. In fact, it was shown in [46] that there is a non-vanishing gap between the exact repair trade-off curve and the functional repair trade-off curve. Hence, functional repair has been studied in [47–49] in an attempt to construct regenerating codes achieving a trade-off close to the optimal while incurring low overheads. Despite these works, a general construction of functional regenerating codes with low overheads, achieving all the points on the trade-off curve between storage capacity and repair bandwidth, has not been achieved. Chapter 3 addresses this problem by proposing an intuitive and practical construction to achieve functional repair for many admissible parameters not achieved in existing literature, and for achieving all the points on the trade-off curve. We employ a geometric view of a linear repair code, and frame the relationship between the local intersection properties of the subspaces spanned by the packets stored by each node, and the global independence properties of those subspaces. We notice and utilize a direct analogy

between information-theoretic identities and identities associated with the intersection and the direct sum of vector spaces, to prove the sufficiency of the proposed conditions. Along with a concise parametric representation of the expression for the points on the trade-off curve, these techniques provide us the necessary conditions to propose a code construction to achieve the complete trade-off curve, with small overheads and low computational complexity.

1.1.3 Secure distributed matrix computation

Data stored on the edge network, while being sensitive in nature, must inevitably be moved around frequently to powerful computation servers for processing. In Chapter 4, we consider the problem of secure distributed computations on private data over the edge network. In the era of big data, performing computationally intensive operations on massive datasets on a single machine is infeasible, and clients often rely on powerful distributed servers for demanding computations. Clients can request computationally expensive tasks to be performed on massive datasets, potentially generated at multiple geographically distributed locations, using special purpose computing servers (eg., Amazon Web Services (AWS), Microsoft Azure, Google Cloud). To speed up the computation, the computation is split into smaller tasks, each of which is carried out by a different server in parallel. While distributed computing provides significant flexibility and speed up, it also leads to growing privacy concerns as the corporations that provide computation services also provide many other digital services, and have access to unprecedented amounts of private user data. If the servers are untrustworthy and the data contains sensitive information, the data sources may be unwilling to share their data due to privacy concerns, thus also slowing the progress of research and development. Therefore, algorithms that would allow users to benefit from powerful untrustworthy servers while keeping their data private are of significant interest. For example, the user client may be a research scientist who aims to perform computations, like linear regression, on patient data from multiple hospitals, where each hospital wants to keep the data of its patients private, and the patient data is made available in the form of matrices. The user should be able to query a function of the datasets hosted by the hospital servers, where the computations are done by third party powerful computation

servers, like AWS, Google Cloud, or Microsoft Azure. The patient data must remain private from the third party servers, and also, the user (research scientist) should not learn anything about the patient data beyond the computation result itself. Moreover, the data sources (hospitals) may be geographically separated, therefore, the communication cost of uploading the data from the hospitals to the servers may be prohibitive. We particularly focus on matrix computations, since matrix computations are fundamental building blocks of a wide range of signal processing and machine learning algorithms. Our goal in this part of the dissertation is to design efficient secure distributed matrix computation (SDMC) algorithms, including matrix addition, matrix multiplication, matrix exponentiation, and matrix inverse, which keep data private from the potentially colluding computing servers as well as the entities requesting the computations, while incurring low communication costs.

The cryptography community has extensively studied the related problem of secure multi-party computation (MPC), also known as secure function evaluation, in which Alice and Bob, having inputs x and y, respectively, want to compute a function $f(x, y)$ jointly, without any of them learning anything about the other's input either from the communication, or from the result of the computation [50]. The secure distributed matrix multiplication (SDMM) problem is related to MPC yet different; the design has to ensure that no computing server learns anything about the original data, but we can decide which part of the data is revealed to each server and in what form. Fully homomorphic encryption (FHE), and "somewhat homomorphic encryption" (SHE) are a class of cryptographic schemes that allow computations on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on plaintext [51–53]. The security of homomorphic encryption schemes depends on the assumed (or proven) computational hardness of problems in ideal lattices.

There is also a growing literature on distributed matrix multiplication, where a lot of effort has been put into speeding up computations, increasing reliability, and/or reducing communication overhead using coding and communication theoretic ideas [54–60].

The initial papers considered a slightly different context of speeding up parallel computations by introducing "computation redundancy" to mitigate the problem of straggling servers [54,55]. Straggling servers refer to slow/unresponsive servers due to which completion of the computation is delayed. A standard way of dealing with stragglers is to introduce "computation redundancy", that is, assigning extra computations to each server. The common theme in the 'coded computation' literature is to treat stragglers as 'erasures' in communications, and apply ideas for coding against erasures, which allows reliable reconstruction of the desired result from an arbitrary set of successfully received symbols. Reference [56] uses polynomial codes to construct a scheme in which the computation is completed as long as any K out of N evaluations of a polynomial are received from the servers. To multiply two matrices $\mathbf{A}$ and $\mathbf{B}$ with the help of N servers, the polynomial code in [56] partitions $\mathbf{A}$ row-wise and $\mathbf{B}$ column-wise (row-by-column partitioning), and generates encoded matrices as evaluations of a polynomial with the blocks as the coefficients, similarly to Reed Solomon codes [61]. In coded computation against stragglers, the performance metric is the *recovery threshold*; that is, the minimum number of successful (non-delayed, non-faulty) servers required for completing the computation.

Subsequent papers, inspired by the works on straggler mitigation in distributed matrix multiplication, consider the SDMM problem from an information theoretic perspective. These papers aim for information theoretic security, independent of the computational capacities of the attackers, as opposed to cryptographic techniques. These works typically assume that the data to be used for the computation is available at the user, and hence the main concern is the privacy against the servers. The earlier papers on SDMM consider *download rate* as the performance metric, which is defined as the ratio of the number of bits required to represent the computation result to the total number of bits that the servers must transmit to the user. Reference [62] uses the idea of polynomial codes from the literature on straggler mitigation to propose an SDMM scheme based on Shamir's secret sharing scheme [63], which is shown to achieve the optimal download rate for *one-sided* SDMM (where only one of the matrices is kept secure). An achievable scheme is also proposed for *two-sided* SDMM (both matrices are kept secure). Reference [64] introduces GASP codes, which improve the download

rate for two-sided SDMM by aligning the degrees of the terms in the polynomial code so that the desired products appear as distinct terms.

In [65], new converse bounds on the optimal download rate for SDMM are obtained by showing that the capacity of a multi-message X-secure T-private information retrieval (MM-XSTPIR) problem ([66, 67]) provides an upper bound on the download rate of the SDMM problem. Reference [60] combines straggler mitigation and secure computation using the batch multiplication approach. Byzantine security is also considered, which refers to security against adversarial servers that may actively corrupt the results they send back to the user. Their scheme is based on Lagrange polynomials, and it achieves the optimal recovery threshold for any multi-linear function computation. Since the scheme in [60] is designed for batch computation of any function, we can adapt it for matrix multiplication using a matrix-partitioning based approach, and compare its performance with that of other matrix partitioning based schemes. This can be done by treating the partitions of the matrices to be multiplied as batches of data. Therefore, if $\mathbf{A}$ is partitioned column-wise and $\mathbf{B}$ is partitioned row-wise into K partitions each, the scheme in [60] has the recovery threshold of $2(K+T-1)+S+2A+1$, where S and A are the numbers of stragglers and Byzantine adversaries, respectively. If $\mathbf{A}$ is partitioned row-wise and $\mathbf{B}$ is partitioned column-wise, into K and L partitions respectively, the recovery threshold is $2(KL+T-1)+S+2A+1$, which is the same performance as that of the GASP codes in [64] for big T, which can be seen by setting $S = A = 0$. For arbitrary matrix partitions, the Lagrange coded scheme does not perform as well as the secure generalized PolyDot (SGPD) codes introduced later in [68]. In [68], the trade-off between the download rate and the recovery threshold, first studied in [69] using the MatDot and PolyDot schemes for straggler mitigation, is extended to SDMM. The SGPD codes achieve the same recovery threshold as Lagrange codes for the sum-of-outer-products method.

The work that is most related to ours is [70], which considers the setting in which the data is generated at distributed source nodes, and does not belong to the user requesting the computation. It extends the BGW (Ben-Or, Goldwasser and Widgerson) scheme from [71], which was first proposed in the context of secure MPC, for multiplication of matrices using a connected network of computing servers. The sources are

assumed not to be connected with each other, while the servers are. The latter assumption is exploited to reduce the communication to the user by allowing the servers to cooperate securely. Thus, the servers share their results from the first round of computation among each other using Shamir's secret sharing scheme, and compute a linear combination of the received shares of the results. This inter-server cooperation allows a smaller number of servers to send these linear combinations to the user. This particular model also imposes privacy against the user, that is, the user cannot learn anything about the data beyond what it learns from the computation result. This constraint is not imposed in other papers, where the user is the source of the input data; and neither in [65], where distributed source nodes generate the data.

With respect to the rich literature on the topic that we have summarized above, our main novel contributions described in this chapter can be summarized as follows:

- We first introduce a novel polynomial coding scheme exploiting the properties of discrete Fourier transform, and show that it achieves a near optimal upload cost for SDMM of two matrices, while achieving the optimal upload cost for the special case of computation with own data, in which the user has access to the matrices used in the computations.
- The proposed scheme can be implemented in an efficient manner using the recently developed fast Fourier transform (FFT) algorithm on finite fields [72], and has negligible decoding complexity.
- We generalize the proposed scheme for SDMM of two matrices to introduce straggler mitigation.
- We extend the proposed scheme to multiply multiple matrices on distributed servers securely, while satisfying the security constraints for keeping the intermediate and final results secure against any T colluding servers. Our scheme has a significantly lower upload cost than the existing schemes, and is naturally scalable to multiplication of an arbitrary number of matrices.

- We describe secure distributed computation schemes for other operations such as matrix addition, matrix transpose, matrix exponentiation, changing the parameters of the secret shares, and solving linear systems, which includes computing the matrix inverse. Thus, it is shown that arbitrary matrix polynomials can be computed securely on distributed servers.

1.2 Publications

In the following, we mention the publications that have resulted from the work presented in this dissertation.

Chapter 2

The content of this chapter has been published in the following:

- N. Mital, D. Gündüz and C. Ling, "Coded caching in a multi-server system with random topology,", IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, 2018, pp 1-6. **(Best student paper award)**
- N. Mital, D. Gündüz and C. Ling, "Coded Caching in a Multi-Server System With Random Topology," in IEEE Transactions on Communications, vol. 68, no. 8, pp. 4620-4631, Aug. 2020.

Chapter 3

Part of the content has been published in the following:

- N. Mital, K. Kralevska, C. Ling and D. Gündüz, "Storage-Repair Bandwidth Trade-off for Wireless Caching with Partial Failure and Broadcast Repair," 2018 IEEE Information Theory Workshop (ITW), Guangzhou, 2018, pp. 1-5.

We then extend the analysis to present a novel code construction to achieve the fundamental limits derived earlier in this chapter.

- N. Mital, K. Kralevska, C. Ling and D. Gündüz, "Practical Functional Regenerating Codes for Broadcast Repair of Multiple Nodes," 2019 IEEE International Symposium on Information Theory (ISIT), Paris, France, 2019, pp. 221-225.
- N.Mital, K.Kralevska, C. Ling and D. Gündüz, "Functional Repair of Multiple Partial Failures with Broadcast: Achieving the Entire Trade-off", (to be submitted).

Chapter 4

The content of this chapter has been submitted for review at the time of writing this thesis:

- N. Mital, C. Ling and D. Gündüz, "Secure Distributed Matrix Computation with Discrete Fourier Transform," arxiv eprints, 2020, under review in IEEE Transactions on Information Theory.

Chapter 2

Coded Caching in a Multi-Server System

2.1 Introduction

Coded caching and distributed storage have received significant attention in recent years to exploit the available memory space and processing power of individual network nodes to increase the throughput and efficiency of data availability. With proactive caching, part of the data can be pushed to nodes' local cache memories during off-peak hours, called the *placement phase*, to reduce the burden on the network during peak traffic periods, called the *delivery phase* [3–14]. Coded caching refers to careful coded or uncoded placement of cache contents in the placement phase, and coded delivery on a multicast channel during the delivery phase. It was shown by Maddah Ali et. al. [3] that coded delivery on multicast links to cache-equipped users helps to leverage a *global caching gain* which provides significant improvement in transmission efficiency. A different type of coded caching also improves the delivery performance in the so-called "femtocaching" scenario [6], where multiple cache-equipped small-cell base stations (SBSs) collaboratively deliver contents to users. Coding for distributed storage systems has been extensively studied in the literature (see, for example, [15]), and in the femtocaching scenario, ideal maximum distance separable (MDS) codes allow users to recover contents by collecting parity bits from only a subset of SBSs they connect to [6].

In this chapter, we combine distributed storage at the SBSs, similar to the "femtocaching" framework [6], with cache storage at the users, and consider coded delivery over error-free shared broadcast links [4]. We consider a library of N files stored across

P SBSs, each equipped with a limited-capacity storage space (see Fig. 2.2). Unlike the existing literature, we consider a boolean random connectivity model [7]: during the delivery phase, each user connects only to a random subset of ρ SBSs, where $\rho \leq P$. This may be due to the density of distribution of SBSs, physical variations in the channel, or due to resource constraints. Most importantly, the connections that form the network topology are not known in advance during the placement phase; therefore, the cache placement cannot be designed for a particular network topology. Storing the files across multiple SBSs, and allowing users to connect randomly to a subset of them results in a loss in multicasting opportunities for the servers, indicating a trade-off between the coded caching gain and the flexibility provided by distributed storage across the servers, which, to the best of our knowledge, has not been studied before.

On the other hand, the presence of multiple servers may improve the latency if user requests can be satisfied in parallel. Accordingly, two scenarios are discussed depending on the delivery protocol. If the servers transmit *successively*, i.e., time-division transmission, the total latency is the sum of the latencies on each link in delivering all the requests. If the servers operate in parallel, then the latency is given by the link with the maximum latency.

We propose a practical coded storage and delivery scheme that exploits MDS coded storage across servers simultaneously with coded caching and delivery to users. In the successive transmission scenario, we show that the cost of the flexibility of distributed storage is a scaling of the latency by a constant. We also characterize the average worst-case latency (over all user-server associations) of the proposed scheme by assuming that the users connect to a uniformly random subset of the servers; and show that it is relatively close to the best-case performance, which is the single-server centralized delivery latency derived in [3], achieved when all the users connect to the same set of servers, maximizing the multicasting opportunities. We observe that, as the server storage capacities increase, the average delivery latency vs. user cache memory trade-off improves, approaching the single-server performance. We give an analytical expression to compute the average delivery latency for different server storage capacities, which is shown to give a fairly accurate estimate of the expected delivery latency when the number of servers is large. We then consider the delivery latency when the

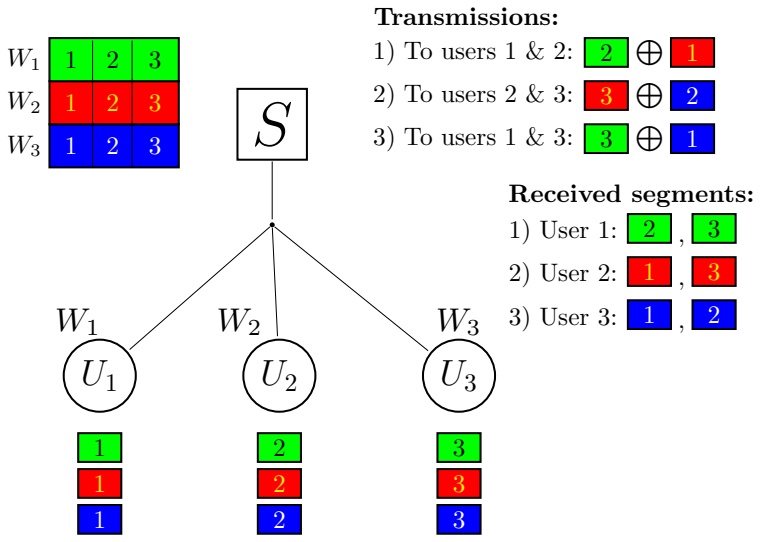


FIGURE 2.1: Maddah-Ali-Niesen (MAN) scheme for 3 users requesting files $\{W_1, W_2, W_3\}$ respectively, each of size F bits, and having cache capacity of $M_U = F$ bits. Each file is divided into 3 segments of size $\frac{F}{3}$ each. Each user caches one segment of each file, thus satisfying the cache constraint. All the requests are satisfied with 3 transmissions, i.e., $3 \times \frac{F}{3} = F$ bits transmitted on the channel.

servers can transmit in parallel. We characterize the achievable average worst case delivery latency of the proposed coded storage and delivery scheme as a function of the server storage capacity for different ρ values.

2.1.1 Choice of connectivity parameter ρ

The parameter ρ depends on various factors, like the multihoming capability of the user devices, density of the Poisson point process according to which one assumes the distribution of SBSs [7], the typical number of available SBSs with good connectivity to a particular user from an arbitrary location, which may be affected by complex environments consisting of buildings and moving objects that can cause physical variations in the channel.

2.1.2 Centralized coded caching with single server

First we briefly describe the Maddah Ali and Niesen (MAN) scheme proposed in the seminal work of [3], which has formed the basis of most schemes developed in later research on coded caching. We use a toy example to explain the essential ingredients

of the scheme (see Fig. 2.1). Consider a library of 3 files, each of size F bits, stored in a centralized server that is connected through a shared link to 3 users. Each user is equipped with a cache memory of size $M = F$ bits. The coded delivery scheme illustrated in Fig. 2.1 satisfies all the requests with 3 coded transmissions, that is, F bits transmitted on the channel, while an uncoded delivery scheme would require 6 transmissions of the segments not in the user caches, that is, $6 \times \frac{F}{3} = 2F$ bits transmitted on the channel.

2.1.3 Coded caching with multiple servers

In a related work [12], the authors study coded caching schemes presented in [3] and [11] when parity servers are available. The authors consider special scenarios with one and two parity servers. They propose a scheme that stripes the files into blocks, and codes them across the servers with a systematic MDS code, and they also propose a scheme for the scenario in which files are stored as whole units in the servers, without striping. In this chapter, we do not specify servers as parity servers, and instead propose a scheme that generalizes to the use of any type of MDS code and any number of storage servers. We study the impact of the topology on the sum and maximum delivery rates, and the trade-off between the server storage capacity and the average of these rates.

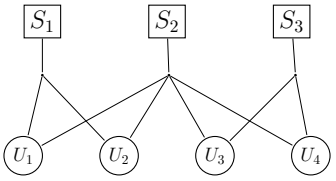
In [13], the authors consider multiple servers, each having access to all the files in the library, serving the users through an intermediate network of relays (see Fig. **??**). They consider the so-called *linear network* model, in which the network topology is fixed but unknown at the relay nodes. The authors study the delivery latency considering parallel transmissions from the servers, and show that there is a gain from using multiple servers when the relay nodes employ simple random linear network coding which makes the network transfer matrix $\mathbf{H}$ rich. Note that, our model considers both limited storage servers and random network topology over the delivery network, which is unknown during the placement phase, but known during the delivery phase. Compared to the linear network model, our model corresponds to an identity network transfer matrix,

in which the scheme of [13] does not provide any gains since it is not optimized for the realization of the topology.

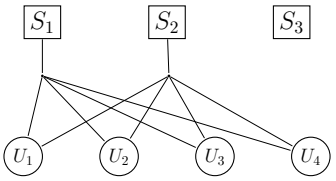
Another line of related works study caching in combination networks [14], [16], which consider a single server serving cache-equipped users through multiple relay nodes (see Fig. **??**). The server is connected to these relays through unicast links, which in turn serve a distinct subset of a fixed number of users through unicast links. A combination network with cache-enabled relay nodes is considered in [16]. In this chapter, we relax the symmetry of a standard combination network and the assumption of a fixed and known network topology, which would be unrealistic in many practical scenarios, to a certain degree by allowing each user to connect to a random fixed number of servers, thus breaking the symmetry from the servers' perspective while maintaining the symmetry from the end-users' perspective.

2.2 Problem Setting

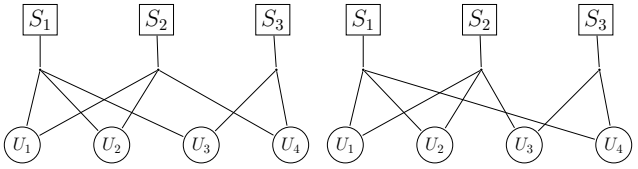
We consider the system model illustrated in Fig. 2.2 with P servers, denoted by $\mathrm{S}_1, \mathrm{S}_2, \ldots, \mathrm{S}_P$, serving K users, denoted by $\mathrm{U}_1, \mathrm{U}_2, \ldots, \mathrm{U}_K$. There is a library of N files $W_1, W_2, \ldots, W_N$, each of length F bits uniformly distributed over $[2^F]$. Each user has access to a local cache memory of capacity $M_U F$ bits, $0 \leq M_U \leq N$, while each server has a storage memory of capacity $M_S F$ bits. The caching scheme consists of two phases: placement phase and delivery phase. We consider a centralized placement scenario as in [3], which is carried out centrally with the knowledge of the servers and the users participating in the delivery phase. However, neither the user demands, nor the network topology is known in advance during the placement phase. In the delivery phase, we assume that each user randomly connects to ρ servers out of P with a uniform distribution over all $\rho-$subsets, where $\rho \leq P$, and requests a single file from the library. This is an instance of a boolean model network [7], which is a first-order approximation of isotropic wireless communication. A user connects to an SBS independently of the other SBSs, where the SBSs are assumed to be uniformly distributed, if the received SNR from that SBS is above a certain threshold, and ρ is



(*a*) $\rho = 2$, $q_1 = 2, q_2 = 4, q_3 = 2$.



(*b*) $\rho = 2$, $q_1 = 4, q_2 = 4, q_3 = 0$ (best topology (for successive transmissions), worst topology (for parallel transmissions)).



(*c*) $\rho = 2, q_1 = 3, q_2 = 3, q_3 = 2$ (worst topology (for successive transmissions), best topology (for parallel transmissions))

FIGURE 2.2: Examples of different network topologies for $P = 3$ and $K = 4$ with $\rho = 2$.

determined by the density of SBSs or other resource constraints. We define $\alpha \triangleq \frac{\rho}{P}$ as the *connectivity* of the network, where $0 \leq \alpha \leq 1$. For $j \in [K]$, let $\mathcal{Z}_j$ denote the set of servers U_j connects to, where $|\mathcal{Z}_j| = \rho$, and $d_j \in [N]$ denotes the index of the file it requests. For example, in Fig. 2.2(a), $\mathcal{Z}_1 = \{\mathrm{S}_1, \mathrm{S}_2\}$, $\mathcal{Z}_2 = \{\mathrm{S}_1, \mathrm{S}_2\}$, $\mathcal{Z}_3 = \{\mathrm{S}_2, \mathrm{S}_3\}$ and $\mathcal{Z}_4 = \{\mathrm{S}_2, \mathrm{S}_3\}$. Let the demand vector be denoted by $\mathbf{d} \triangleq (d_1, d_2, ..., d_K)$. The topology of the network, i.e., which users are connected to which servers, and the demands of the users are revealed to the servers at the beginning of the delivery phase.

The complete library must be stored at the servers in a coded manner to provide redundancy, since each user connects only to a random subset of the servers. Since any user should be able to reconstruct any requested file from its own cache memory and the servers it is connected to, the total cache capacity of a user and any ρ servers must be sufficient to recover the whole library; that is, we must have $M_U + \rho M_S \geq N$.

Let $\mathcal{K}_p$ denote the set of users served by S_p, for $p \in [P]$, and define the random variable $Q_p \triangleq |\mathcal{K}_p|$, which denotes the number of users served by S_p. We shall denote a particular realization of Q_p as q_p and define $\mathbf{q} \triangleq (q_1, \ldots, q_p)$, where we have $\sum_{p=1}^{P} q_p = K\rho$.

For example, in Fig. 2.2(a), we have $\mathcal{K}_1 = \{\mathrm{U}_1, \mathrm{U}_2\}, \mathcal{K}_2 = \{\mathrm{U}_1, \mathrm{U}_2, \mathrm{U}_3, \mathrm{U}_4\}, \mathcal{K}_3 = \{\mathrm{U}_3, \mathrm{U}_4\}$, and $\mathbf{q} = (2, 4, 2)$. In the delivery phase, server S_p transmits message X_p of size R_pF bits to the users connected to it, i.e., the users in set $\mathcal{K}_p$, over the corresponding shared link. We assume that each server is allocated a separate orthogonal delivery channel, and the message it transmits is received by all the users connected to this server. The message X_p is a function of the demand vector $\mathbf{d}$, the network topology, the storage contents of server S_p, and the cache contents of the users in $\mathcal{K}_p$. User U_k receives the messages $X_{\mathcal{Z}_k} \triangleq \{X_p : p \in \mathcal{Z}_k\}$, and reconstructs its requested file W_{d_k} using these messages and its local cache contents.

2.2.1 Formal problem statement

We now provide the formal definition of the caching problem. Let $\{W_n\}_{n=1}^N$ be N independent random variables each uniformly distributed over $[2^F]$ for some $F \in \mathbb{N}$. Each W_n represents a file of size F bits. Let $R_p, p \in [P]$, be the number of bits, normalized by the size of a file, transmitted by server $p \in [P]$ during the delivery phase. A $(M_S, M_U, R_1, \ldots, R_P)$ storage and caching scheme consists of P server storage functions, K caching functions, $P\binom{P}{\rho}^K$ encoding functions, and $K\binom{P}{\rho}^K$ decoding functions.

The caching function

$$\phi_k : [2^F]^N \to [2^{\lfloor FM_U \rfloor}], \quad k \in [K], \tag{2.1}$$

maps the library $\{W_n\}_{n=1}^N$ into the cache contents, of user U_k during the placement phase:

$$V_k \triangleq \phi_k(W_1, \ldots, W_N) \tag{2.2}$$

The server storage function

$$\sigma_p : [2^F]^N \to [2^{\lfloor FM_S \rfloor}], \quad p \in [P] \tag{2.3}$$

maps the library $\{W_n\}_{n=1}^N$ into the storage of server S_p:

$$Y_p \triangleq \sigma_p(W_1, \ldots, W_N). \tag{2.4}$$

We define a separate encoding function for each server depending on the network topology. Hence, the encoding function for server $S_p, p \in [P]$,

$$\psi^p_{\{\mathcal{K}_p\}_{p=1}^P} : [N]^K \times [2^{\lfloor FM_S \rfloor}] \to [2^{\lfloor FR_p \rfloor}] \tag{2.5}$$

maps the demand vector and the memory contents of server S_p to message X_p, i.e.,

$$X_p \triangleq \psi^p_{\{\mathcal{K}_p\}_{p=1}^P}(\mathbf{d}, Y_p), \tag{2.6}$$

which is delivered to the users in $\mathcal{K}_p$ during the delivery phase. Finally, we define a separate decoding function for each user depending on the network topology. Hence, the decoding function for user $U_k, k \in [K]$, is

$$\mu^k_{(\{\mathcal{Z}_k\}_{k=1}^K)} : [N]^K \times [2^{\lfloor FR_{\pi^k(1)} \rfloor}] \times \cdots \times [2^{\lfloor FR_{\pi^k(\rho)} \rfloor}] \times [2^{\lfloor FM_U \rfloor}] \to [2^{\lfloor F \rfloor}], \tag{2.7}$$

where $\pi^k(1), \ldots, \pi^k(\rho)$ denote the ρ servers in set $\mathcal{Z}_k$, maps the demand vector $\mathbf{d}$, the received signals $X_{\mathcal{Z}_k}$ from the servers in $\mathcal{Z}_k$, and the local cache content V_k to the estimate $\hat{W}_{d_k}$, i.e.,

$$\hat{W}_{d_k} \triangleq \mu^k_{(\{\mathcal{Z}_k\}_{k=1}^K)}(\mathbf{d}, X_{\mathcal{Z}_k}, V_k) \tag{2.8}$$

The probability of error for this scheme, for a fixed topology, is defined as

$$\max_{\mathbf{d} \in [N]^K} \max_{k \in [K]} \Pr(\hat{W}_{d_k} \neq W_{d_k}). \tag{2.9}$$

We remark here that the storage and caching functions σ_p and ϕ_k do not depend on the network topology, while the encoding and decoding functions do.

Definition 2.1. The tuple $(M_S, M_U, R_1, \ldots, R_P)$ is said to be achievable if for every $\epsilon > 0$ and large enough file size F there exists a $(M_S, M_U, R_1, \ldots, R_P)$ caching scheme with probability of error less than ϵ.

Our goal is to minimize the delivery latency, which is the time by which all the user requests can be satisfied. Among other parameters, delivery latency also depends on the operation of the SBSs. If each SBS transmits over an orthogonal frequency band, the requests can be delivered in parallel, and the delivery latency is given by $T_{pd} = \max_p R_p$. If, instead, the servers transmit successively in a time-division manner, which is suitable for user devices that are simple and not capable of multihoming on multiple frequencies, the normalized delivery latency will be given by $T_{sd} = \sum_{p=1}^{P} R_p$. Our goal will be to find the average worst-case delivery latency, where the worst case refers to the fact that all the users can correctly decode their requested files, independent of the combination of files requested by them, and the averaging is over all possible network topologies. Assuming that $N \geq K$ (i.e., the number of files is larger than the number of users), it is not difficult to see that all the users requesting a different file corresponds to the worst-case scenario. We would also like to remark that, under uniform file popularity, the probability of experiencing this worst-case demand distribution increases significantly with N, and approaches 1 for N values that one expects to experience in practice.

2.3 Coded Distributed Storage and Caching Scheme

We first note that our system model brings together aspects of distributed storage and proactive caching/coded delivery. To see this, consider the system without any user caches, i.e., $M_U = 0$, which is equivalent to a distributed storage system with unreliable servers, where random $P - \rho$ out of P servers are inactive. It is known that MDS codes provide much higher reliability and efficiency compared to replication in this scenario [15]. On the other hand, when the servers are reliable, i.e., $\rho = P$, our system is equivalent to the one in [3], and coded delivery provides significant reductions in the delivery latency. Accordingly, our proposed scheme brings together benefits from coded storage and coded delivery. To illustrate the main ingredients of the proposed scheme we assume $M_S = \frac{N}{\rho}$ in this section, and extend to other server capacities in later sections.

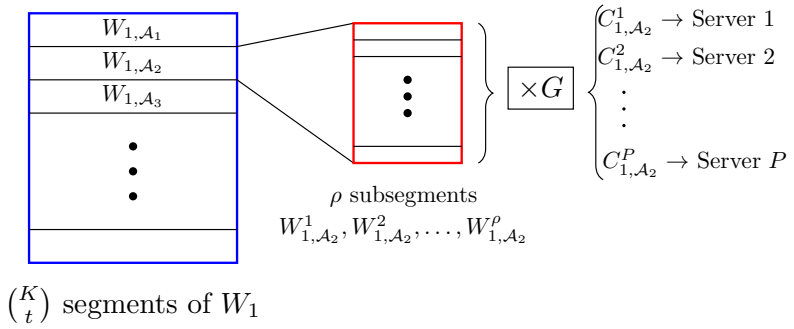


FIGURE 2.3: Segmentation, MDS coding and placement of files.

2.3.1 Server storage placement

We first describe how the files are stored across the SBS servers in order to guarantee that each user request can be satisfied from any ρ servers a user may connect to (see Fig. 2.3). We define $t \triangleq \frac{KM_U}{N}$, and assume initially that t is an integer, i.e., $t \in [0 : M_U]$. The solution for non-integer t values will be obtained through memory-sharing [3]. Each file is divided into $\binom{K}{t}$ equal-size non-overlapping segments. We enumerate them according to distinct t-element subsets of $[K]$, where $W_{j,\mathcal{A}}$ denotes the segment of W_j that corresponds to subset $\mathcal{A}$. We have $W_j = \bigcup_{\mathcal{A}\subset[K]:|\mathcal{A}|=t} W_{j,\mathcal{A}}$, $j \in [N]$.

Each segment is further divided into ρ equal-size non-overlapping sub-segments denoted by $W^l_{j,\mathcal{A}}$, $l \in [\rho]$. The ρ sub-segments of each segment are coded together using a (P, ρ) linear MDS code with generator matrix G, giving as output P coded subsegments for segment $W_{j,\mathcal{A}}$, denoted by $C^l_{j,\mathcal{A}}, l \in [P]$. $C^l_{j,\mathcal{A}}$ is a linear combination of the subsegments of the segment corresponding to subset $\mathcal{A}$, of file W_j. $C^l_{j,\mathcal{A}}$ will be stored in server S_l, $\forall l \in [P], j \in [N]$, and $\mathcal{A} \subset [K], |\mathcal{A}| = t$. Since each sub-segment is of length $\frac{F}{\rho\binom{K}{t}}$, every linear combination $C^l_{j,\mathcal{A}}$ is of the same length; and hence, server storage capacity constraint of $M_S F = \frac{NF}{\rho}$ is met with equality.

Remark 2.1. *We assume that each user knows the generator matrix of the MDS code to be able to reconstruct any coded subsegment $C^l_{j,\mathcal{A}}$ from the uncoded segment $W_{j,\mathcal{A}}$.*

2.3.2 User cache placement

Using the placement scheme proposed in [3] for user caches, each segment of a file, $W_{j,\mathcal{A}}$, is placed into the caches of all the users U_k for which $k \in \mathcal{A}$, i.e., each user caches $\binom{K-1}{t-1}$ segments of each file, or $\frac{\binom{K-1}{t-1}}{\binom{K}{t}}NF = \frac{t}{K}N = M_U F$ bits, meeting the user cache capacity constraint.

2.3.3 Delivery phase

We first make the following observation about the above placement scheme: in the worst-case demand scenario, consider any $t+1$ users. Any t out of these $t+1$ users share in their caches one segment of the file requested by the remaining user. Enumerate these subsets of $t+1$ users as $\mathcal{H}_i$, $i \in \left[\binom{K}{t+1}\right]$. Consider server S_p, $p \in [P]$, and one of the q_p users connected to it, say U_k. Then, for any subset $\mathcal{H}_i$, that includes k, i.e., $k \in \mathcal{H}_i$, the segment $W_{d_k,\mathcal{H}_i\setminus\{k\}}$ is needed by user U_k, but is not available in its cache because $k \notin \mathcal{H}_i \setminus \{k\}$, while it is available in the caches of the users in $\mathcal{K}_p \bigcap \mathcal{H}_i \setminus \{k\}$. The MDS coded subsegment of $W_{d_k,\mathcal{H}_i\setminus\{k\}}$ stored by S_p is $C^p_{d_k,\mathcal{H}_i\setminus\{k\}}$, and since the users know the generator matrix G, each user which has $W_{d_k,\mathcal{H}_i\setminus\{k\}}$ in its cache can reconstruct $C^p_{d_k,\mathcal{H}_i\setminus\{k\}}$ as well. Then, for each $\mathcal{H}_i$ that includes at least one user from $\mathcal{K}_p$, S_p transmits

$$X_p(\mathcal{H}_i) = \bigoplus_{k\in\mathcal{K}_p \bigcap \mathcal{H}_i\setminus\{k\}} C^p_{d_k,\mathcal{H}_i\setminus\{k\}}, \tag{2.10}$$

where $\bigoplus$ denotes the bitwise XOR operation. Then, $\left|\left\{i \in \left[\binom{K}{t+1}\right] : k \in \mathcal{H}_i\right\}\right| = \binom{K-1}{t}$ is the number of messages transmitted by server S_p that contain the coded version of a segment requested by U_k, and is also equal to the number of segments of W_{d_k} not present in the cache of user U_k. Overall, the message transmitted by S_p is given by

$$X_p = \bigcup_{i\in\left[\binom{K}{t+1}\right]:\mathcal{K}_p \bigcap \mathcal{H}_i\neq\phi} X_p(\mathcal{H}_i). \tag{2.11}$$

From the transmitted message $X_p(\mathcal{H}_i)$ in (2.10) for each set $\mathcal{H}_i$, user U_k can decode the MDS coded version $C^p_{d_k,\mathcal{H}_i\setminus\{k\}}$ of its requested segment $W_{d_k,\mathcal{H}_i\setminus\{k\}}$. With the

transmissions from all the servers, U_k receives ρ coded versions of each missing segment from the ρ servers it is connected to. Since each segment is coded with a (P, ρ) MDS code, the user is able to decode each missing segment of its request.

Note that each transmitted message $X_p(\mathcal{H}_i)$ by a server is of length $F/\rho\binom{K}{t}$ bits. The number of messages transmitted by S_p is

$$\left|\left\{i \in \left[\binom{K}{t+1}\right] : \mathcal{K}_p \bigcap \mathcal{H}_i \neq \phi\right\}\right| \tag{2.12}$$

$$= \binom{K}{t+1} - \left|\left\{i \in \left[\binom{K}{t+1}\right] : \mathcal{K}_p \bigcap \mathcal{H}_i = \phi\right\}\right| \tag{2.13}$$

$$= \binom{K}{t+1} - \binom{K-q_p}{t+1}. \tag{2.14}$$

That is, server S_p transmits a total of $R_p = {}^{F}/_{\rho\binom{K}{t}} \left[\binom{K}{t+1} - \binom{K-q_p}{t+1}\right]$ bits.

The delivery latency performance of this proposed coded storage and delivery scheme with both successive and parallel SBS transmissions will be studied in the following two sections.

Remark 2.2. *Due to the symmetry in the network across servers and users, the delivery latency of this scheme depends only on the* $\mathbf{q}$ *vector, not the particular network topology, i.e., what matters is the number of users served by each server, not the identity of the users. More specifically, all permutations of a* $\mathbf{q}$ *vector, and the associated users, result in the same latency. Hence, we define the "type" of a network topology as a vector of dimension* $K+1$, $\mathbf{g}$, *where* g_i *denotes the number of servers serving* i *users, for* $i = 0, 1, \ldots, K$. *We have* $0 \leq g_i \leq P$, $\sum_{i=0}^{K} g_i = P$ *and* $\sum_{i=0}^{K} i g_i = K\rho$.

2.4 Successive SBS Transmissions

In this section we assume that the SBSs share the same communication resources, and hence, transmit successively to avoid interference. When the SBSs transmit successively in time, the normalized delivery latency is given by

$$T_{sd} \triangleq \sum_{p=1}^{P} R_p = \frac{1}{\rho\binom{K}{t}} \sum_{p=1}^{P} \left[\binom{K}{t+1} - \binom{K-q_p}{t+1}\right] \tag{2.15}$$

$$= \frac{1}{\alpha}\frac{(K-t)}{(t+1)} - \frac{1}{\rho\binom{K}{t}}\sum_{p=1}^{P}\binom{K-q_p}{t+1} \tag{2.16}$$

$$= \frac{1}{\alpha}\frac{(K-t)}{(t+1)} - \frac{1}{\rho\binom{K}{t}}\sum_{i=0}^{K} g_i\binom{K-i}{t+1}. \tag{2.17}$$

To characterize the "best" and "worst" network topologies that lead to the minimum and maximum delivery latency, respectively, we present the following lemma without proof.

Lemma 2.1. *For $n_1, n_2, r \in \mathbb{Z}^+$ satisfying $r \leq n_1$ and $n_1 + 2 \leq n_2$, we have*

$$\binom{n_1}{r} + \binom{n_2}{r} \geq \binom{n_1+1}{r} + \binom{n_2-1}{r}. \tag{2.18}$$

The lemma above indicates the "convex" nature of the binomial coefficients in (2.16); that is, the points $(r, \binom{r}{r})$, $(r+1, \binom{r+1}{r})$, $\ldots$, $(n_1+n_2-r, \binom{n_1+n_2-r}{r})$ form a convex region. From Lemma 2.1, it can be deduced that the second summation term in (2.16) takes its minimum when $\max_p(q_p) \leq \min_p(q_p) + 1$, $p \in [P]$, i.e., the values of q_p are as close to each other as possible. This corresponds to the class of topologies with the highest delivery latency (see Fig. 2.2(c) for an example). The topology that requires the minimum delivery latency of $T_{sd} = \frac{K-t}{t+1}$ is when q_p is either 0 or K for each server, or equivalently, when all the users are connected to the same ρ servers (see Fig. 2.2(b) for an example).

Next we study the average worst-case normalized delivery latency, where the average is taken over all possible network topologies. As we have seen above, the delivery latency depends on the topology, and for a given topology, the "worst-case" delivery latency refers to the worst-case demand combination when each user requests a different file. Note that, in the worst case, due to the symmetry in the network and the proposed caching and delivery scheme, the latency depends only on the type of the network topology. We further assume that the probability of having any network of the same type is the same.

Lemma 2.2. *Let w_i be the probability of exactly i users being served by a server; that is, $w_i = Pr\{q_p = i\}, p \in [P]$. We have*

$$\mathbb{E}[g_i] = w_i P. \tag{2.19}$$

Proof. The number of servers serving exactly i users, g_i, can be written as

$$g_i = \sum_{p=1}^{P} \mathbb{1}_{\{q_p=i\}}. \tag{2.20}$$

Taking expectation on both sides, we have

$$\mathbb{E}[g_i] = \sum_{p=1}^{P} \Pr\{q_p = i\} \tag{2.21}$$

$$= w_i P. \tag{2.22}$$

□

The following theorem presents the average normalized worst-case delivery latency of the proposed scheme under successive transmissions, which follows by taking the expectation of both sides of Eq. (2.17) and Lemma 2.2.

Theorem 2.2. *The average worst-case normalized delivery latency of the proposed scheme over all topologies under random user-server association is given by*

$$\mathbb{E}[T_{sd}] = \frac{1}{\alpha}\frac{(K-t)}{(t+1)} - \frac{1}{\alpha\binom{K}{t}} \sum_{i=0}^{K} w_i \binom{K-i}{t+1}. \tag{2.23}$$

Since we have assumed uniform random connectivity, we have $w_i = \binom{K}{i}\binom{P-1}{\rho-1}^i \binom{P-1}{\rho}^{K-i} / \binom{P}{\rho}^K = \binom{K}{i}\alpha^i(1-\alpha)^{K-i}$. The average worst-case latency is given in the following corollary. The average worst-case normalized delivery latency with successive transmissions under uniformly random user-server association is given by

$$\mathbb{E}[T_{sd}] = \frac{K-t}{t+1}\left[\frac{1-(1-\alpha)^{t+1}}{\alpha}\right]. \tag{2.24}$$

Proof. By plugging in $w_i = \binom{K}{i}\alpha^i(1-\alpha)^{K-i}$ in Eq. (2.23), we obtain the above simplified expression. □

2.4.1 Redundancy in server storage capacity

In the analysis above, we have set the server storage capacity to $M_S = \frac{N}{\rho}$. On the other hand, for a given user cache capacity M_U, the minimum server storage capacity that would allow the reconstruction of any demand combination is given by $M_S = \frac{N-M_U}{\rho}$. In this case, we cache the same $\frac{M_U}{N}$ fraction of the library in all the user caches during the placement phase, and deliver the remaining fraction of the demands from the servers, which is identical to the scheme in [16] when the user and its connected servers have just enough space to store the entire library. The worst-case delivery latency in this case is given by $T_{sd} = K\left(1-\frac{M_U}{N}\right) = K - t$.

Next, we consider the case when there is redundancy in server memories; that is, $\frac{N}{\rho} < M_S \leq N$. Assume that $M_S = \frac{N}{\rho - z}$ for some integer $z \in [\rho - 1]$. Define $\hat{\alpha} \triangleq \frac{\rho - z}{P}$. Since α is defined as the connectivity of the network, $\alpha - \hat{\alpha}$ is the storage redundancy. For non-integer values of z, the solution can be obtained by memory-sharing.

In this case, a $(P, \rho - z)$ MDS code is used for server storage placement, allowing each user to reconstruct any requested file by connecting to $\rho - z$ servers. The user cache placement is done as in the previous section. In the delivery phase, each user randomly connects to ρ servers. We now have a degree of freedom thanks to the additional storage space available at each server. Each user can obtain a segment from any $\rho - z$ of the ρ servers it is connected to by receiving one coded subsegment from each of them. The choice of the servers that deliver the coded subsegments to the users is made such that the multicasting opportunities across the network are maximized. We construct an incidence matrix A of dimensions $P \times K$ such that $a_{ij} = 1$ if S_i is connected to U_j, $a_{ij} = 0$ otherwise. Consider the $(t+1)-$element subset $\mathcal{H}_i$, and the file segments $W_{d_k, \mathcal{H}_i \setminus \{k\}}, \forall k \in \mathcal{H}_i$. Consider the columns of A corresponding to the users in $\mathcal{H}_i$ and the matrix Q formed by them. Define the minimum cover of $\mathcal{H}_i$ as the smallest l for which a $l \times (t+1)$ submatrix of Q has at least $\rho - z$ non-zero values in each column. The servers corresponding to the l rows of this submatrix have to transmit

$$
\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \end{array}
\begin{array}{c}
\begin{array}{ccccc} U_1 & U_2 & U_3 & U_4 & U_5 \end{array} \\
\left[\begin{array}{ccccc}
1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1
\end{array}\right]
\end{array}
$$

FIGURE 2.4: An example 7×5 incidence matrix ($P = 7, K = 5$) with $\rho = 4$.

one coded message each to satisfy the requests for the missing segments corresponding to $\mathcal{H}_i$. Therefore, the total number of transmissions required to deliver the segments $W_{d_k,\mathcal{H}_i\setminus\{k\}}, k \in \mathcal{H}_i$, is equal to the minimum cover of $\mathcal{H}_i$.

As an example, consider the incidence matrix as shown in Fig. 2.4, which corresponds to a system with $P = 7$ servers and $K = 5$ users, where each user connects to $\rho = 4$ servers. Assume that the server storage capacity is $M_S = \frac{N}{\rho-2}$ and $t = 1$. In this setting, coded subsegments of requested files can be delivered to $t + 1 = 2$ users through multicasting, and it is sufficient for each user to receive coded segments from $\rho - 2 = 2$ servers. Then, for the user set $\mathcal{H}_i = \{1, 2\}$, we consider the submatrix corresponding to the columns 1 and 2 and rows 1 and 2 (marked by the blue dashed lines in Fig. 2.4), which is the smallest submatrix satisfying the condition that each column has at least $\rho - z = 2$ 1s. Hence, the minimum cover for $\mathcal{H}_i = \{1, 2\}$ is equal to the number of rows of this submatrix, that is, 2. For $\mathcal{H}_i = \{3, 4\}$ (marked by the red dashed lines in Fig. 2.4), the minimum cover is 3. Thus, from (2.10), for segments $W_{d_k,\{3,4\}\setminus\{k\}}, k \in \{3, 4\}$, S_3 transmits the message $X_3(\{3, 4\}) = \bigoplus_{k\in\{3,4\}} C^3_{d_k,\{3,4\}\setminus\{k\}}$, S_4 transmits $X_4(\{3, 4\}) = C^4_{d_3,\{4\}}$, and S_5 transmits $X_5(\{3, 4\}) = C^5_{d_4,\{3\}}$. The total number of transmissions is 3. We can go through all the $(t+1)-$ element subsets of the users and identify for each of them the minimum cover. We note that in the successive transmission scenario, the total latency does not depend on the server transmitting each subsegment, since the contribution to the total latency is the same. In the above example servers S_1 and S_6 could also deliver the two coded subsegments to users U_1

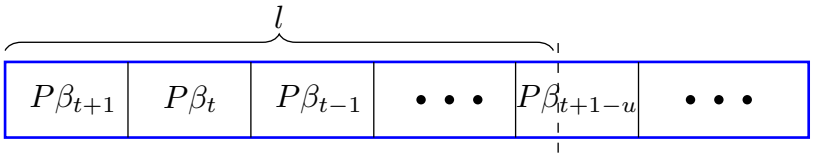


FIGURE 2.5: The ordering of servers to count the minimum cover. The dashed line indicates the point at which enough servers have been counted to deliver $\hat{\alpha}$ coded subsegments to all users in $\mathcal{H}$.

and U_2. The selection of the servers matters in the case of parallel transmissions.

2.4.2 Performance analysis

In this section, we derive an analytical expression for the expected delivery latency in the asymptotic regime, i.e., when $P \to \infty$, while α and $\hat{\alpha}$ are fixed. Consider a particular subset $\mathcal{H}$ of $t+1$ users. Define β_i as the fraction of servers serving i users in $\mathcal{H}$, $i = 0, 1, \ldots, t+1$. Thus, we have

$$\beta_i = \frac{1}{P} \sum_{p=1}^{P} \mathbb{1}_{\{|\mathcal{H} \cap \mathcal{K}_p| = i\}}. \tag{2.25}$$

Taking expectation on both sides of Eq. (2.25), we have

$$\mathbb{E}[\beta_i] = \frac{1}{P} \sum_{p=1}^{P} \mathbb{E}[\mathbb{1}_{\{|\mathcal{H} \cap \mathcal{K}_p| = i\}}] \tag{2.26}$$

$$= \frac{1}{P} \sum_{p=1}^{P} \Pr(|\mathcal{H} \cap \mathcal{K}_p| = i) \tag{2.27}$$

$$= \Pr(|\mathcal{H} \cap \mathcal{K}_p| = i) \tag{2.28}$$

$$= \binom{t+1}{i} \alpha^i (1-\alpha)^{(t+1-i)}, \tag{2.29}$$

where (2.28) follows due to the symmetry across all the servers. By the law of large numbers, $\beta_i \to \mathbb{E}[\beta_i]$ for all $i \in [K]$, as $P \to \infty$. Also, the topology becomes symmetric across all users as $P \to \infty$, i.e., almost all user subsets of the same size are served by the same number of servers. We group the servers serving the same number of users and arrange them in the order as illustrated in Fig. 2.5, where the first $P\beta_{t+1}$ servers serve $t+1$ users in $\mathcal{H}$, the next $P\beta_t$ servers serve exactly t users in $\mathcal{H}$, and so on. To

compute the minimum cover l, i.e., the minimum number of servers that are needed to deliver $\hat{\alpha}$ coded subsegments to each user in $\mathcal{H}$, we start counting from the left, until each user in $\mathcal{H}$ collects $\hat{\alpha}$ coded subsegments. For some $u \in [0:t]$, we count till the $(u+1)$-th set of servers which serve $t+1-u$ users in $\mathcal{H}$, as in Fig. 2.5. When counting the set of servers serving $t+1-u$ users, note that, according to our scheme, the $t+1-u$ users can each extract one coded subsegment from a message transmitted by a server in that set. Therefore, $\lceil \frac{t+1}{t+1-u} \rceil$ servers are required to serve one coded subsegment each to the $t+1$ users in $\mathcal{H}$. Define δ as the number of coded subsegments required by a single user in $\mathcal{H}$ from the set of servers serving $t+1-u$ users. Therefore, for $P \to \infty$, the minimum cover can be written as

$$l \approx P \sum_{j=0}^{u-1} \beta_{t+1-j} + \delta \left\lceil \frac{t+1}{t+1-u} \right\rceil \tag{2.30}$$

for some $u \in [0:t]$, where (2.30) follows thanks to the symmetry across users. Note that the above analysis is asymptotic, and does not hold in general for a finite P. Since a message transmitted by a server serving i users in $\mathcal{H}$ delivers i coded subsegments in total to the i users, the total number of coded subsegments delivered by the l servers that form the minimum cover for the users in $\mathcal{H}$ must be at least $(t+1)\hat{\alpha}$; that is,

$$P \sum_{j=0}^{u-1} (t+1-j)\beta_{t+1-j} + \delta' \left\lceil \frac{t+1}{t+1-u} \right\rceil \geq (t+1)P\hat{\alpha}, \tag{2.31}$$

where $\delta' \triangleq (t+1-u)\delta$. The value of u is determined by solving for

$$\begin{aligned} 0 \leq (t+1)P\hat{\alpha} - P \sum_{j=0}^{u-1} (t+1-j)\beta_{t+1-j} \\ \leq (t+1-u)\beta_{t+1-u}. \end{aligned} \tag{2.32}$$

From Eq. (2.29) and the asymptotic convergence of β_i to its expectation, we have

$$\begin{aligned} &\sum_{j=0}^{u-1} (t+1-j)\beta_{t+1-j} \\ &\qquad \overset{P\to\infty}{\to} \sum_{j=0}^{u-1} \binom{t+1}{t+1-j} (t+1-j)\alpha^{(t+1-j)}(1-\alpha)^j \end{aligned}$$

$$=\alpha^{t+1}(t+1)\sum_{j=0}^{u-1}\binom{t}{s}\left(\frac{\alpha}{1-\alpha}\right)^{-j} \tag{2.33}$$

We substitute (2.33) into (2.32) to solve for u. Having first determined u from Eq. (2.32), and then δ from (2.31), we can find the minimum cover l from Eq. (2.30) for $P \to \infty$. The delivery latency can thus be estimated as

$$\mathbb{E}[T_{sd}] = \frac{1}{(\rho - z)}\left(\frac{K-t}{t+1}\right) l, \tag{2.34}$$

where the factor $\frac{1}{(\rho-z)}\left(\frac{K-t}{t+1}\right)$ is obtained by multiplying the normalized size of each coded subsegment, given by $\frac{1}{(\rho-z)\binom{K}{t}}$, with the number of $(t+1)-$user subsets, given by $\binom{K}{t+1}$. It will be seen in Section 2.7 that Eq. (2.34) provides a fairly accurate estimate of the expected delivery latency when the number of servers P is large.

2.5 Lower Bound

In this section, we derive a tight lower bound on the minimum expected delivery latency with uncoded cache placement, coded distributed storage in the servers, and successive transmissions, which shows the optimality of the caching and delivery scheme proposed in Section 2.4 in certain regimes. Following [73], we will first represent the problem as a set of index coding problems.

In the index coding problem [74], a sender wishes to communicate an independent message $M_j, j \in [B]$, uniformly distributed over $[2^{nr_j}]$, to the j^{th} user among B users by broadcasting a message X^n of length n. Each user j knows a subset of the messages targeting these B users, indicated by $\mathcal{B}_j, \mathcal{B}_j \subset \{M_1, \ldots, M_B\}$, referred to as side information. A rate tuple $(r_1, \ldots, r_B)$ is achievable, for large enough n, if every user can restore its desired message with high probability based on X^n and its side information. The index coding problem can be represented as a directed graph G with B nodes, where node i represents message M_i, and a directed edge connects node i to node j if user j knows message M_i as side information. For our problem setting, where we have the file library $\{W_i\}_{i=1}^N$, each file $W_i, i \in [N]$, of size F bits is divided into 2^K non-overlapping segments denoted by $W_{i,\mathcal{A}}$, $\mathcal{A} \in 2^{[K]}$, where $2^{[K]}$ indicates the

power set $\{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}\ , \ldots, [K]\}$. The segment $W_{i,\mathcal{A}}$ denotes the part of file W_i cached exclusively by users in set $\mathcal{A}$. This is the most general representation of an uncoded caching scheme at the users. For each demand vector $\mathbf{d}$ with distinct requests, corresponding to the worst case scenario, we consider an index coding problem with $K2^{K-1}$ independent messages, each of which represents a segment requested by a particular user and cached by a different subset of the remaining users.

We generate a directed graph with $K2^{K-1}$ nodes corresponding to these messages, such that, for $i \neq j$ and $\mathcal{A}_i \subset [K] \setminus \{i\}$ and $\mathcal{A}_j \subset [K] \setminus \{j\}$, there is a directed edge from node $W_{d_i,\mathcal{A}_i}$ to $W_{d_j,\mathcal{A}_j}$, $i \neq j$, if user U_j caches the segment $W_{d_i,\mathcal{A}_i}$; that is, if $j \in \mathcal{A}_i$. In the single server centralized setting, we get a lower bound using the index coding bound [75]. Multi-server index coding has been studied as the distributed index coding problem in [76], [77]. In the distributed index coding problem, the servers are considered to store a subset of the messages in uncoded form, and each user is connected to all the servers, whereas in our problem each user can connect to ρ out of P servers randomly. Therefore, for the user to be able to retrieve any requested file from the servers it connects to, the files must be stored using a distributed storage scheme in the servers. Therefore, we analyse the case where the files are stored using erasure codes in the servers. In that, we encode the segment $W_{d_i,\mathcal{A}}$ into P distinct coded subsegments, denoted by $(C^1_{d_i,\mathcal{A}}, \ldots, C^P_{d_i,\mathcal{A}}) \in \prod_{p\in[P]} \left[2^{n_p r^p_{d_i,\mathcal{A}}}\right]$, where $n_p = FR_p$ is the length of message in bits transmitted by server S_p, such that any ρ coded subsegments can be used to reconstruct the original segment. $r^p_{d_i,\mathcal{A}}$ is the rate at which server S_p transmits the coded subsegment $C^p_{d_i,\mathcal{A}}$ corresponding to user $U_i's$ request, and we have $\sum_{j=1}^{\rho} n_{\pi(j)} r^{\pi(j)}_{d_i,\mathcal{A}} \geq |W_{d_i,\mathcal{A}}|$ as a necessary condition to ensure that the segment $|W_{d_i,\mathcal{A}}|$ can be reconstructed by receiving any ρ distinct coded subsegments, where $\pi(j), j \in [\rho]$, are the ρ servers in set $\mathcal{Z}_i$. Recall that $\mathcal{Z}_i$ is the set of ρ servers that serve user U_i. Also note that we do not code across files, but encode each file separately.

For the multi-server scenario, we consider P index coding problems, each represented as a distinct directed graph $G_p, p \in [P]$. Each node in G_p corresponds to a distinct coded subsegment $C^p_{d_i,\mathcal{A}}$, which is requested by user U_i and available in server S_p. By distinct coded subsegments we mean that $H(W_{d_i,\mathcal{A}}|C^p_{d_i,\mathcal{A}}, C^q_{d_i,\mathcal{A}}) < H(W_{d_i,\mathcal{A}}|C^p_{d_i,\mathcal{A}})$ for all $p, q \in [P], p \neq q$. G_p has the same structure as G, with the subsegments requested

by users not served by server S_p removed. Let the set of nodes in the index coding problem represented by graph G_p be denoted by $\mathcal{I}_p$.

We have the following multi-server index coding bound applying the result in [75] separately on each of the P index coding problems.

Theorem 2.3. *If the rate tuple* $\{r^1_{1,\mathcal{A}}, \ldots, r^1_{K,\mathcal{A}}, \ldots, r^p_{1,\mathcal{A}}, \ldots, r^p_{K,\mathcal{A}}, \ldots, r^P_{1,\mathcal{A}}, \ldots, r^P_{K,\mathcal{A}}\}_{\mathcal{A}\subseteq[K]}$ *is achievable for the multi-server index coding problem represented by the set of directed graphs* $G_p, p = 1, \ldots, P$, *under the constraint* $\sum_{j=1}^{\rho} n_{\pi(j)} r^{\pi(j)}_{d_i,\mathcal{A}} \geq |W_{d_i,\mathcal{A}}|$, *and inter-file coding is not allowed, then* $r^p_{j,\mathcal{A}} = 0$ *if server* S_p *does not serve user* U_j, *and*

$$\sum_{p=1}^{P} \sum_{\mathcal{J}_p} r^p_{j,\mathcal{A}} \leq 1 \tag{2.35}$$

for all $\mathcal{J}_p \subseteq \mathcal{I}_p$ *where the subgraph of* G_p *over* $\mathcal{J}_p$ *does not contain a directed cycle.*

Remark 2.3. *Theorem 2.3 holds when the nodes in the P index coding problems correspond to distinct coded subsegments, that is, there are no repeating nodes in any two index coding problems. Distributed storage schemes which concatenate repetition codes with other storage codes may not satisfy the bound in Theorem 2.3 (for example, see [78]). References [76] and [77] may indicate how to compute the capacity under such distributed storage schemes, but they are outside the scope of this dissertation.*

Remark 2.4. *There are non-MDS distributed storage codes, called regenerating codes, that utilize increased storage capacity on the servers to reduce the repair bandwidth [15]. Theorem 2.3 holds for them unless some repetition code is used, because the problem can still be represented as P independent index coding problems. For example, Theorem 2.3 holds if a product matrix code [28] is used for distributed storage. However, a sub-optimal delivery latency is achieved, because each server stores a larger number of packets that have to be transmitted to the connected users for successful file reconstruction.*

To identify the acyclic sets $\mathcal{J}_p$ in the subgraph G_p, consider the permutations $\mathbf{u} = (u_1, \ldots, u_K)$ of $[K]$. To determine the tightest bound, we may only consider the largest such sets without a directed cycle. For a given $\mathbf{u}$, the largest set of nodes not containing

a directed cycle is

$$\left\{C^p_{d_{u_i},\mathcal{A}_i} : \mathcal{A}_i \subseteq [1:K] \setminus \{u_1,\ldots,u_i\}, i = 1,\ldots,K\right\}.$$

Each permutation $\mathbf{u}$ gives a unique acyclic set of nodes of the graph. The subsegment $C^p_{d_i,\phi}$ is not cached in any user, so there is no outgoing edge from $C^p_{d_i,\phi}$ to any other nodes in any sub-index coding problem. Therefore $C^p_{d_i,\phi}$ is always in the set $\mathcal{J}_p$.

Consider first $M_S = \frac{N}{\rho}$. In that case, $\sum_{j=1}^{\rho} n_{\pi(j)} r^{\pi(j)}_{d_i,\mathcal{A}} = \sum_{j=1}^{\rho} |C^{\pi(j)}_{d_i,\mathcal{A}}| = |W_{d_i,\mathcal{A}}|$. Following Theorem 2.3, in order to recover all the desired segments for each user, the deliver latency, T_{sd} must satisfy

$$\begin{aligned} FT_{sd} \geq \sum_{p=1}^{P} \Bigg(& \sum_{\mathcal{A}\subseteq[1:K]\setminus\{u_1\}} |C^p_{d_{u_1},\mathcal{A}}| \\ & + \cdots + \sum_{\mathcal{A}\subseteq[1:K]\setminus(\{u^i\}\cap\mathcal{K}_p)} |C^p_{d_{u_i},\mathcal{A}}| + \cdots + \sum_{\mathcal{A}\subseteq[1:K]\setminus(\{u^K\}\cap\mathcal{K}_p)} |C^p_{d_{u_K},\mathcal{A}}| \Bigg) \qquad (2.36) \\ \text{s.t.} \quad & \sum_{p\in\mathcal{Z}_j} |C^p_{d_{u_j},\mathcal{A}}| = |W_{d_{u_j},\mathcal{A}}| \qquad j \in [K], \qquad (2.37) \end{aligned}$$

for every permutation $\mathbf{u}$, and for every network topology.

We have $|C^p_{d,\mathcal{A}}| = \frac{|W_{d,\mathcal{A}}|}{\rho}$ for $M_S = \frac{N}{\rho}$, due to (P,ρ) MDS coded storage. In Eq. (2.36), in the summation for a fixed value of p, the number of terms with $|\mathcal{A}| = i$ is $\binom{K}{i+1} - \binom{K-q_p}{i+1}$. Thus we have

$$\begin{aligned} FT_{sd} \geq & \sum_{p=1}^{P}\sum_{i=0}^{K-1} \frac{\left(\binom{K}{i+1} - \binom{K-q_p}{i+1}\right)}{\binom{K}{i}} x_i^p \qquad (2.38) \\ = & \sum_{p=1}^{P}\sum_{i=0}^{K-1} \frac{\left(\binom{K}{i+1} - \binom{K-q_p}{i+1}\right)}{\rho\binom{K}{i}} x_i \qquad (2.39) \\ = & \sum_{i=0}^{K-1} \left[\sum_{p=1}^{P} \frac{\left(\binom{K}{i+1} - \binom{K-q_p}{i+1}\right)}{\rho\binom{K}{i}}\right] x_i \qquad (2.40) \\ \text{while} \quad & x_0 + x_1 + \cdots + x_K \geq F, \qquad (2.41) \\ \text{and} \quad & x_1 + 2x_2 + \cdots + Kx_K \leq \frac{KM_U}{N} F \qquad (2.42) \end{aligned}$$

where $x_i \triangleq \sum_{\mathcal{A}\subset[K]:|\mathcal{A}|=i}|W_{j,\mathcal{A}}| = \binom{K}{i}|W_{j,\mathcal{A}}| = \rho\binom{K}{i}|C^p_{j,\mathcal{A}}|$ is the total normalized size of all segments of file j cached by i users; or equivalently, $x^p_i \triangleq \binom{K}{i}|C^p_{j,\mathcal{A}}| = \frac{1}{\rho}x_i$ is the total normalized size of all subsegments of file j cached by i users and stored in server S_p. We minimize the lower bound in Eq. (2.40) over all segment sizes x_i, which is a linear program with two linear constraints (2.41) and (2.42), where the former follows from the sum of all fractions of the files being one, while the latter follows from the user cache memory constraint. The solution of a linear program lies on one of the corner points of the feasible region. The feasible region defined by the constraints has only one corner point characterized by

$$x_i = \begin{cases} F & i = t, t = \frac{KM_U}{N} \\ 0 & \text{otherwise} \end{cases}.$$

Therefore, Eq. (2.40) simplifies to

$$T_{sd} \geq \sum_{p=1}^{P} \frac{\left(\binom{K}{t+1} - \binom{K-q_p}{t+1}\right)}{\rho\binom{K}{t}}, \tag{2.43}$$

which is achieved by our delivery scheme. This proves the optimality of the delivery scheme for successive transmission proposed in Section 2.4 under the assumption of MDS coded storage at the servers and uncoded caching at the users.

2.5.1 Redundancy in server storage

When there is redundant server storage capacity, i.e., server storage capacity is $M_S = \frac{N}{\rho - z}$, consider the constraint $\sum_{j=1}^{\rho} n_{\pi(j)} r^{\pi(j)}_{d_i,\mathcal{A}} \geq |W_{d_i,\mathcal{A}}|$ in Theorem 2.3. Since the bound in Theorem 2.3 is a linear program of the rates of transmission of the coded subsegments from the servers, the optimal solution lies on one of the corner points of the feasible region defined by the constraint. The corner points for $M_S = \frac{N}{\rho - z}, z \in [\rho - 2]$, are those where $n_{\pi(j)} r^{\pi(j)}_{d_i,\mathcal{A}} = \frac{|W_{d_i,\mathcal{A}}|}{\rho - z}$ for all $j \in \mathcal{R}, \mathcal{R} \subset \mathcal{Z}_i, |\mathcal{R}| = \rho - z$, and equal to 0 for all $j \in \mathcal{Z}_i \setminus \mathcal{R}$. The optimal solution should lie on the corner point which chooses $\mathcal{R}$ such that the servers in $\mathcal{R}$ have the most multicasting opportunities, and can thus deliver $\rho - z$ coded subsegments of the requested segments to the users in the

minimum number of transmissions. This is equivalent to finding the minimum cover for each multicast group as described in Section 2.4.1.

When fractional repetition (FR) codes are used for server storage [79], the minimum cover scheme may not be optimal. However, since FR codes have a maximum code rate of $\frac{1}{2}$, we cannot have distributed storage schemes where $M_S \leq \frac{2N}{P}$. Thus the minimum cover scheme is optimal for server storage capacities $M_S \leq \frac{2N}{P}$. We illustrate with a toy example that the bound in Theorem 2.3 does not hold when FR codes are used.

Toy example:

Consider the simple scenario with $P = 2$ servers, $K = 3$ users illustrated in Fig. 2.6, where we assume each server can store all the $N = 3$ files, i.e., $M_S = 3$, and each user has cache capacity $M_U = 1$. Let the cache contents of U_1, U_2, U_3 be W_2, W_3, W_1, respectively, and the demand vector $\mathbf{d} = \{W_1, W_2, W_3\}$. In this example, the demands can be satisfied by S_1 transmitting $W_1 \oplus W_2$, and S_2 transmitting $W_1 \oplus W_3$, that is, the delivery latency of $T_{sd} = 2$ is achievable. However, Theorem 2.3 gives the bound on delivery latency as $T_{sd} \geq 3$. U_2 receives its requested file W_2 with added interference of W_1 from S_1, which it cannot remove using its cache contents. However, U_2 adds the messages from S_1 and S_2 to align the interference on W_2 with W_3, which it can remove by using its cache contents, thus doing a sort of interference alignment. In contrast, if MDS coded storage were used, the interference alignment type of scheme would not be possible due to distinct coded subsegments transmitted by both servers.

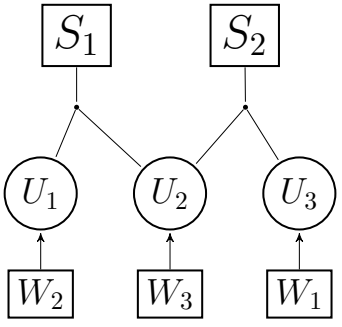


FIGURE 2.6: Toy example with $P = 2, K = N = M_S = 3, M_U = 1$

The polymatroidal capacity region for multi-server index coding has been characterized in [76] for full user-server connectivity and uncoded server storage. Characterizing

the capacity region for partial user-server connectivity, and constructing an optimal joint server storage and caching scheme for FR coded distributed storage is an interesting open problem for future work.

2.6 Parallel SBS Transmissions

When SBSs can deliver in parallel without interfering with each other, the normalized delivery latency is dictated by the SBS that has to deliver the maximum number of bits:

$$T_{pd} \triangleq \max_{q_p} \frac{1}{\rho\binom{K}{t}} \left[\binom{K}{t+1} - \binom{K-q_p}{t+1} \right]. \tag{2.44}$$

The "best" and "worst" network topologies in the parallel transmission scenario are different from those in the successive transmission scenario. The most balanced topology, i.e., the one with the minimum value of the maximum q_p has the "best" (lowest) delivery latency, contrary to the successive transmission scenario, in which this would be the "worst" topology. The corresponding delivery latency can be obtained by substituting $q_p = \lceil \frac{K\rho}{P} \rceil$ in (2.44). The topology with the maximum possible q_p, i.e., any topology with at least one server connected to all K users, is the "worst" topology since it has the highest delivery latency.

2.6.1 Redundant server storage capacity

The minimum server storage capacity that would allow the reconstruction of any demand combination is given by $M_S = \frac{N-M_U}{\rho}$. In this case, we cache the same $\frac{M_U}{N}$ fraction of the library in all the user caches during the placement phase, and deliver the remaining fraction of the demands from the servers without multicasting. The worst-case delivery latency in this case is $T_{pd} = \frac{K}{\rho}\left(1 - \frac{M_U}{N}\right)$.

Next, we consider the case when there is redundancy in server memories; that is, $\frac{N}{\rho} < M_S \leq N$. Assume that $M_S = \frac{N}{\rho - z}$ for some integer $z \in [\rho - 1]$. For non-integer values of z, the solution can be obtained by memory-sharing. Notice that, as for successive

transmissions, users can select the servers from which to receive coded subsegments. A greedy server allocation algorithm is used. The algorithm assigns the multicast messages to the servers trying to keep the number of messages delivered by each server as evenly distributed as possible. At any point in time, if a server has delivered a higher number of messages than all the other servers, even if a better multicasting opportunity is available to this server, that server is not assigned a multicast message in order to balance the number of messages delivered by each server in a greedy manner. Instead, the server with the next best multicasting opportunity and a smaller count of transmissions is assigned to transmit a particular coded subsegment to a multicast group. Compare this with the algorithm for successive transmission, where a multicast message is always assigned to the server with the maximum multicasting opportunity. It is easy to see that the delivery latency achieved depends on the order in which the algorithm assigns multicast messages to the servers. Thus the proposed algorithm is suboptimal. Numerical results illustrating the performance of the proposed delivery algorithm will be presented in the next section.

2.7 Results and Discussions

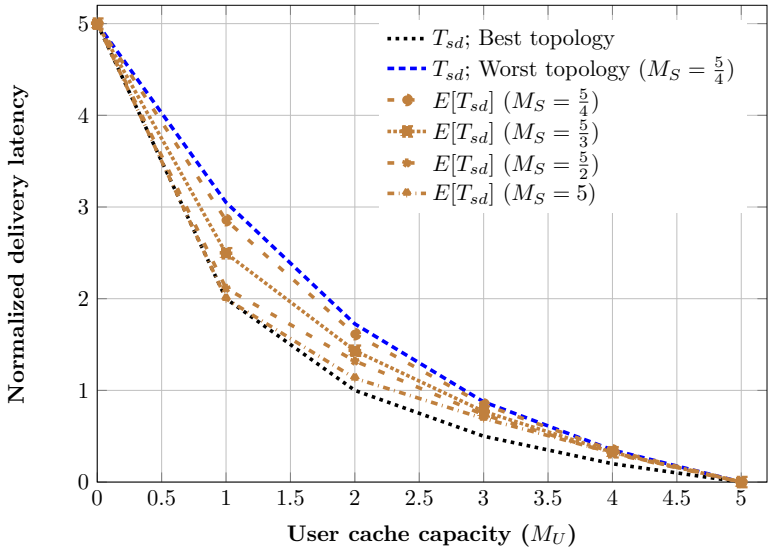


FIGURE 2.7: Average normalized delivery latency vs. user cache capacity M_U, for $P = 7, N = K = 5, \rho = 4$, and for server storage capacities of $M_S = \frac{5}{4}, \frac{5}{3}, \frac{5}{2}, 5$.

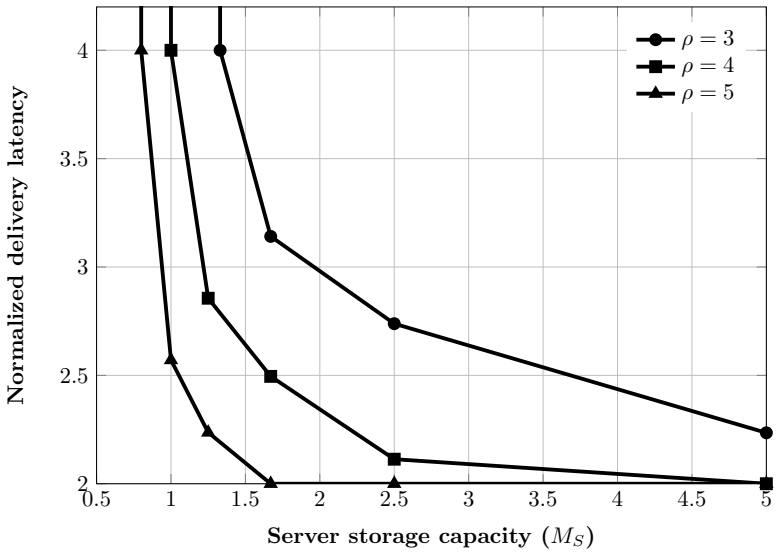


FIGURE 2.8: Average normalized delivery latency vs. server storage capacity M_S, for $P = 7, N = K = 5, M_U = 1$ for successive SBS transmissions.

In Fig. 2.7 we plot the achievable trade-off between the user cache capacity and the normalized delivery latency, T_{sd}, for the best and worst topologies, and the average normalized delivery latency over all topologies, for successive transmission. The trade-off curves are plotted for different server storage capacities. We observe that the gap between the worst and the best topologies can be significant. From (2.17) and (2.23) we can deduce that, for successive transmission the worst topology delivery latency; and hence, the average delivery latency of the proposed scheme are both within a multiplicative factor of $\frac{1}{\alpha}$ of the best topology delivery latency. We observe from Fig. 2.7 that the delivery latency decreases significantly, particularly for low M_U values, as the redundancy in server storage increases.

In Fig. 2.8 the average delivery latency for successive transmission is plotted as a function of the server storage capacity for server storage capacities $M_S \in [\frac{N-M_U}{\rho}, N]$. The figure is obtained by performing Monte Carlo simulations with uniform random realizations of the topology and averaging the delivery latency over them. We observe from Fig. 2.8 that the average delivery latency decreases rapidly for an initial increase in the server storage capacity, which is more significant for high ρ values. This is because, thanks to MDS-coded storage at the servers, the number of available multi-casting opportunities increases with the redundancy across servers. Fig. 2.8 highlights the fact that, for successive delivery and sufficient network connectivity, increasing the

server storage beyond a certain value has little or no impact on the delivery latency.

In Fig. 2.9, it is shown that Eq. (2.34) in Section 2.4.2 gives a fairly accurate estimate of the expected delivery latency for successive transmissions with redundant server storage capacity, especially for small server storage capacities. The theoretical estimate diverges a little from the expected rate for large server storage capacity, before again converging where the delivery latency saturates at the minimum. Also, comparing the plot for $\rho = 9, P = 21$ in Fig. 2.9 with the plot for $\rho = 3, P = 7$ in Fig. 2.8, where the connectivity α is the same, we observe that the average delivery latency decreases faster for $\rho = 9, P = 21$; that is, for larger values of P.

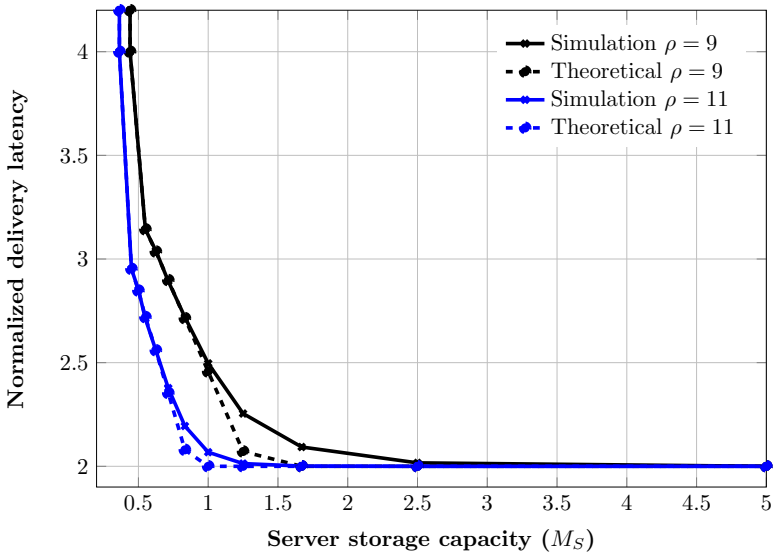


FIGURE 2.9: Comparing the simulation with the theoretical, Average normalized delivery latency vs. server storage capacity, for $P = 21, N = K = 5, M_U = 1$ for successive SBS transmissions.

The average delivery latency for parallel transmissions is plotted with respect to the user cache capacity in Fig. 2.10, using (2.44). We observe as before that increasing the server storage capacity gives significant gains in the average delivery latency, especially for low values of M_U. Unlike the case for successive transmissions, the average delivery latency for $M_U = 0$ also reduces as the server storage capacity is increased.

The average delivery latency for parallel transmissions is plotted with respect to the server storage capacity, M_S, in Fig. 2.11. Unlike the delivery latency for successive transmissions, we can see that the delivery latency does not saturate, and keeps

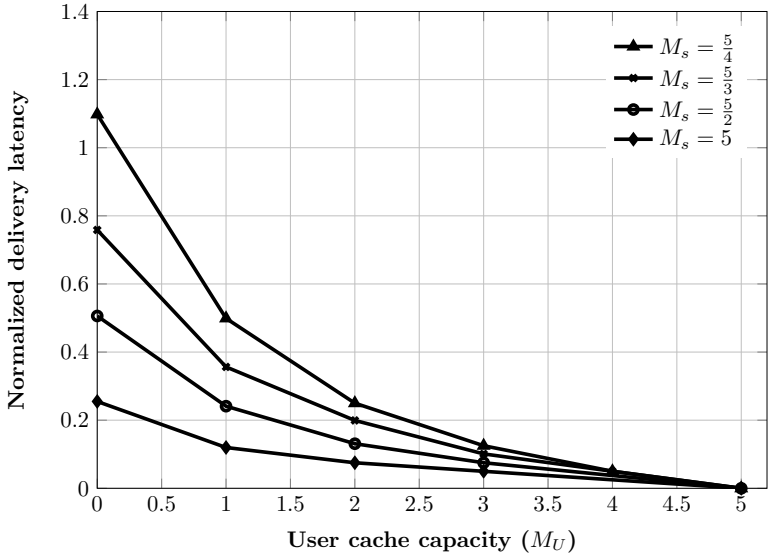


FIGURE 2.10: Average normalized delivery latency for parallel transmissions vs. user cache capacity M_U, for $P = 7, N = K = 5, \rho = 4$, and for server storage capacities of $M_S = \frac{5}{4}, \frac{5}{3}, \frac{5}{2}, 5$.

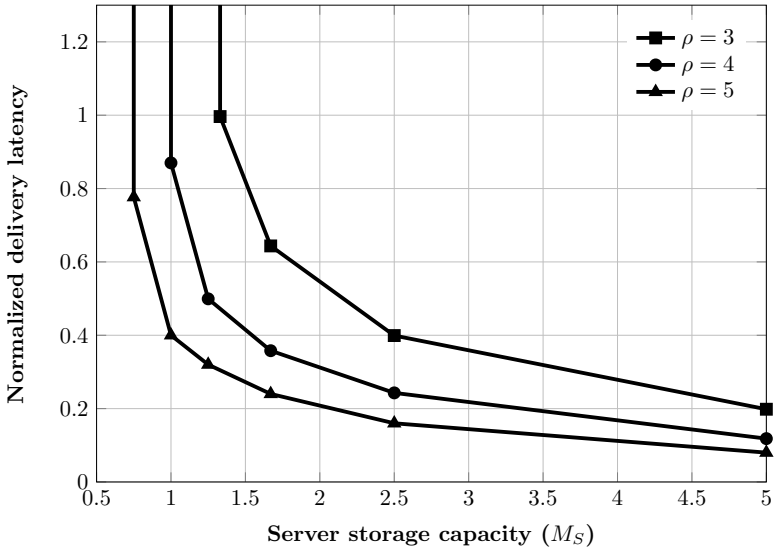


FIGURE 2.11: Average normalized delivery latency vs. server storage capacity M_S, for $P = 7, N = K = 5, M_U = 1$ for parallel transmissions.

decreasing until all the files are stored at each of the servers. We also observe as before that the increase in network connectivity α helps reduce the delivery latency significantly, especially for low server storage capacity M_S.

2.8 Conclusions and Future Work

We have studied a multi-server coded caching and delivery network, in which cache-equipped users connect randomly to a subset of the available servers, each with its own limited storage capacity. While this allows each server to have only a limited amount of storage capacity, it requires coded storage across servers to account for the random topology. We proposed a joint coded storage, caching and delivery scheme that jointly applies MDS-coded storage at the servers, and uncoded caching and coded delivery to the users. The achievable delivery latency of this scheme for both successive and parallel transmissions from the SBSs are presented, with increasing user cache memory as well as increasing server storage capacity, and their averages over random network topologies are plotted. The analysis shows that when the server storage capacity is increased, the delivery latency can be reduced significantly, for both successive transmissions as well as parallel transmissions. However, it is also observed that for sufficient network connectivity, increasing server storage beyond a certain value provides little benefit. Increasing server storage has a more significant impact when there is low connectivity, and when user cache capacities are small.

The scheme proposed in this chapter does not attempt to reduce the file subpacketization that is exponential in the parameters K and t for the proposed scheme, which is based on the MAN scheme, thus allowing only a small number of users to be meaningfully served in practice. Recent works employing Placement Delivery Array (PDA) designs [80–82], and other works [83–85], propose coded caching schemes with subexponential, or polynomial subpacketization, at the cost of lower caching gain. The scheme proposed in this chapter can be adapted to work with low subpacketization coded caching schemes, and it presents an interesting future direction of work to observe whether low subpacketization schemes can provide almost similar gains as the schemes based on the MAN scheme in a distributed setting considered in this chapter.

Chapter 3

Regenerating Codes for Distributed Storage

3.1 Introduction

Caching popular contents closer to end-users, particularly in the available storage space at the wireless network edge, is attracting a lot of attention in the recent years as a promising method to alleviate the increasing traffic on the backhaul links of wireless access points, e.g., small cell base stations, and to improve the quality of service for end users, particularly by reducing the latency [1, 17], or the energy consumption [9]. The literature on distributed caching systems focuses mostly on the code design or the resource allocation for efficient storage of popular contents, assuming reliable cache nodes. However, storage devices are often unreliable and prone to failures; thus, efficient repair techniques that guarantee continuous data availability are essential for a successful implementation of distributed caching and content delivery techniques in practice.

Maximum distance separable (MDS) codes are typically used for distributed caching of contents at multiple access points [1,17,18]. MDS codes provide flexibility for storage so that users with different connectivity or mobility patterns can download a file from only a subset of the access points. In particular, an (n, k) MDS code encodes a file of size M bits by splitting it into k equal-size packets and encoding them into n packets which are stored at n cache nodes. Each data packet consists of a set of symbols in a finite field $\mathbb{F}_q$. The original file can be reconstructed by accessing any k out of n packets from k distinct access points. This property is called the *data reconstruction property*.

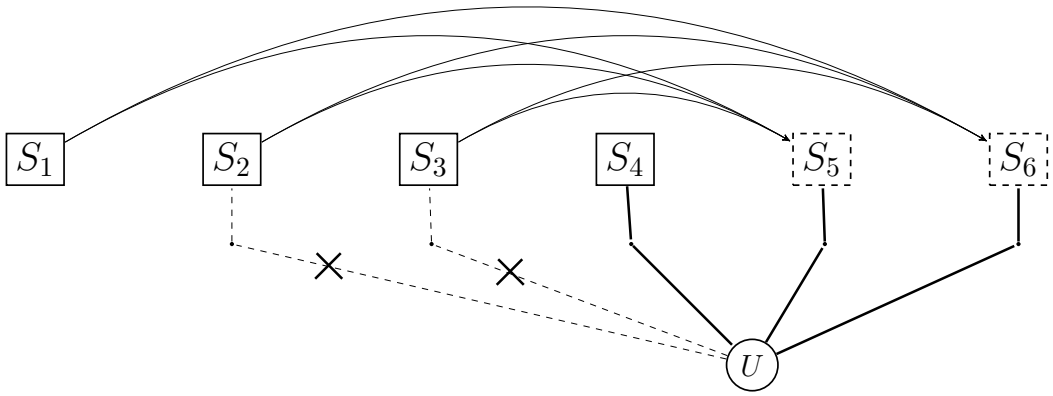


FIGURE 3.1: A distributed SBS cache scenario.

3.1.1 Regenerating codes

When some nodes partially or fully fail, or when content has to be moved to new cache nodes, their cache contents must be regenerated, either in the same failed node, or in the new cache nodes where the content has to be moved, to be able to continue serving users with the same amount of redundancy. An important objective of edge caching in wireless networks is to reduce the backhaul link loads; therefore, we consider *cache recovery at the edge*; that is, rather than updating the failed cache contents from a central server through backhaul links, the failed cache contents are regenerated with the help of surviving cache nodes. This is called the *data regeneration property* of a code. Besides the conventional use-case of repairing failed storage nodes, the repair problem also models the scenario of a network of distributed SBS caches considered in [86], where the user requires $k = 3$ servers to obtain its desired file, but parts of the file are stored only on servers S_1, S_2, S_3 and S_4, while the user has access to servers S_4, S_5 and S_6 (see Fig. 3.1). Hence, instead of servers S_5 and S_6 fetching content from a central server via the backhaul, 3 servers, namely S_1, S_2 and S_3, "repair" the new cache nodes S_5 and S_6, or in other words, move the necessary content to S_5 and S_6 so that the user can obtain the file from S_4, S_5 and S_6. The total amount of data transferred from the surviving nodes to repair the failed nodes is called the *repair bandwidth*. A trivial way to repair the failed nodes is to allow the nodes being repaired to connect to any k surviving nodes, download the entire file, and recover the data that was stored originally. Downloading the entire file to recover the data of a node

that stores only a fraction of the file is wasteful. Conventional MDS codes treat the packets stored in a node as a single symbol belonging to the finite field $\mathbb{F}_q$. It can be shown that when the nodes are only permitted to perform linear operations over $\mathbb{F}_q$ when using conventional MDS codes, the total repair bandwidth can not be smaller than the length of the entire file. Thus, conventional MDS codes have high storage efficiency, but their repair bandwidth is large [15].

In contrast, *regenerating codes* are codes that treat the data stored on each node as a vector of S data packets. In particular, a file of size M bits is split into P data packets consisting of symbols in $\mathbb{F}_q$. A total of S coded packets are stored on each node having a storage capacity of α bits. Linear operations over $\mathbb{F}_q$ in this case permit the transfer of a fraction of the data stored in a single node, thus achieving a lower repair bandwidth than conventional MDS codes. Similarly to existing literature [19–23], we refer to S as subpacketization. Such codes, constructed over a vector alphabet, are called *array codes* in the distributed storage literature. During repair, r failed nodes are allowed to connect to d surviving nodes, and download a total of $\gamma = d\beta$ bits to repair the lost contents. Dimakis *et al.* showed in [15] that there is a fundamental trade-off between the storage capacity α and the repair bandwidth γ by mapping the repair problem in a distributed storage system to a multicasting network coding problem [24] over an information flow graph [25]. The trade-off is shown to be characterized by the cut-set bound for linear network coding, and can be achieved with high probability using random linear coding [24]. The analysis in [15] focuses on a single node repair, that is, losing one node triggers the repair process. Existing literature has mainly focused on two extremes of this trade-off: the minimum-storage regenerating (MSR) point and the minimum-bandwidth regenerating (MBR) point. Apart from a low repair bandwidth, it is also desirable for a regenerating code to have low subpacketization. Subpacketization determines the smallest file size that can be encoded, and the number of operations required in the encoding and decoding process.

3.1.2 Exact vs functional repair

Another terminology frequently found in the literature is that of *exact repair* and *functional repair*. Functional repair refers to the repair process in which the replacement of a set of failed nodes is such that the data reconstruction property and data regeneration property is preserved in the resulting network of n nodes, while the coded content in the replacement nodes may be different from the coded content in the original nodes. In contrast, exact repair refers to the repair process in which the replacement nodes store exactly the same coded content as stored originally in the failed nodes. Exact repair is often preferred over functional repair because the former is predictable and does not require nodes to communicate their changing coding coefficients to all other nodes. However, functional repair is generally more flexible than exact repair.

3.1.3 Related work

Initially, most literature considered the exact repair of a single node at a time [19, 20, 23, 26–28]. In [28], the product-matrix framework for constructing optimal repair MSR and MBR codes is proposed, which achieves the MBR point for all admissible parameters, and achieves the MSR point for parameters restricted to $d \geq 2k-2$. The product-matrix construction employs an optimal (linear) subpacketization in n,k and d. Zig-zag codes proposed in [19] allow arbitrarily high code rates ($k/n \to 1$) for the MSR point but require a subpacketization level that is exponential in k. Other MSR constructions, including Reed Solomon (RS) codes, have been proposed that meet the cut-set bound, but employ an exponential, or at best polynomial, subpacketization [20–23].

It was first observed in [29] that multiple node repair, that is, when the repair process starts only after r nodes fail, is more efficient in terms of the repair bandwidth per node, compared to repairing each node as it fails. In [30] and [31], the authors introduce cooperative regenerating codes, which repair multiple failures cooperatively by allowing each of the r nodes being repaired to collect data from any $n-r$ surviving nodes, and then to cooperate with the other $r-1$ nodes being repaired. In [32], the authors design a minimum storage cooperative regenerating (MSCR) code for $n = 2k, d = 2k-2, r = 2$,

which they generalize to the repair of systematic nodes for $n = 2k, d = n - r, 2 \leq r \leq n - k$ in [33]. An obvious drawback of these constructions is that they restrict to a limited set of parameters. Ye and Barg [34] gave an explicit MSCR construction for all parameters, that is, for $2 \leq r \leq n - k, d > k$, but employ a subpacketization that is exponential in the parameters n, k and d. The product-matrix construction of [28] for single node repairs was extended to MSCR in [35], meeting the cut-set bound with optimal subpacketization, but for parameters restricted to $d \geq 2k - 2 - r$. In [36], the authors gave an explicit construction of minimum bandwidth cooperative regenerating (MBCR) codes for all parameters n, k, d, r, employing only a linear subpacketization.

Instead, similarly to [37,38], we consider broadcast repair; that is, transmissions from each node are received in an error-free manner by all the other nodes. The storage-repair bandwidth trade-off for the repair of multiple fully failed nodes using broadcast repair was derived in [38]. Additionally, we consider the *partial repair* problem, studied in [37], in which a node loses only a part of its contents, and the remainder of the contents may be used along with the transmissions from the surviving nodes to repair the content, thus further reducing the repair bandwidth. In [37], the authors derive a lower bound on the number of packet transmissions at the MSR point for error-free broadcast partial repair, and provide an explicit code construction for a special case. The information flow graph construction in [37] does not capture the relation between storage capacity and the repair bandwidth, thus focusing only on one of the extreme points on the storage-repair bandwidth trade-off curve, corresponding to the MSR point. In this chapter, we obtain the entire optimal trade-off curve.

In [39], the authors present the storage-repair bandwidth trade-off for clustered storage networks, where multiple nodes within a cluster fail. This is close to the partial failure model that we consider since each cluster is equivalent to a node with multiple memory units, and failure of a subset of memory units in a node is the same as partial failure. However, we consider partial failures at multiple nodes, which is equivalent to failure of multiple nodes in multiple clusters, and we also consider broadcast transmissions from the non-failed nodes. Another model studied in the literature is the centralized repair model [40, 41], in which the surviving nodes transmit the repairing packets to a centralized node, which then repairs the failed nodes. The centralized

model does not require the nodes being repaired to exchange data like the cooperative repair model, thus making the repair process simpler; moreover, the storage-repair bandwidth trade-off achieved with centralized repair is better than that with cooperative repair. In [40,42], it is shown that cooperative repair achieves the minimum repair bandwidth of centralized repair under exact repair but at a slightly higher storage cost. In [41], it is shown that the optimal functional MBR point for centralized repair of multiple nodes is not achievable under exact repair. Unlike the centralized repair model, we consider broadcast repair, similarly to [37, 38], that is, transmissions from the surviving nodes are received in an error-free manner by all the nodes being repaired. The broadcast repair model is almost equivalent to the centralized multi-node repair model studied in [40, 41]. Therefore, the codes that we construct for broadcast repair are directly applicable to the centralized repair model. However, the main difference between the broadcast repair model and the centralized repair model is that while centralized repair employs two phases in the repair process, first where a centralized entity receives the transmissions of the surviving nodes, and second where the centralized entity repairs the failed nodes by passing messages to them, broadcast repair employs a single phase of broadcast transmissions from surviving nodes, and therefore, is simpler and more efficient.

Since random linear coding is asymptotically optimal for network coding [43], it is also asymptotically optimal for functional repair in distributed storage [15]. However, random linear coding is not the most practically feasible scheme for distributed storage because of large overheads and high computational complexity of the Gaussian elimination-based decoding [44]. Therefore, most literature on regenerating codes is focused on exact repair due to smaller overheads and predictability of the system when the content of the storage nodes does not change with time. However, it was shown in [45] that a large portion of the functional repair trade-off curve cannot be achieved under exact repair. In fact, it was shown in [46] that there is a non-vanishing gap between the exact repair trade-off curve and the functional repair trade-off curve. Hence, functional repair has been studied in [47–49] in an attempt to construct regenerating codes achieving a trade-off closer to the optimal while incurring low overheads. Hollmann and Poh in [48] viewed a regenerating code as a collection of sets of subspaces

of a vector space, in which reconstruction corresponds to generating the vector space while repair corresponds to generating a subspace. This differs from the purely coding theoretic view where the file is encoded using a generator matrix, like in [28]. With respect to the vector space interpretation of linear functional repair codes characterized by Hollmann and Poh, it is stated in [47] - "The properties of the code are determined entirely by the manner in which the various spaces intersect. The advantage of the geometric perspective is that many classical geometric objects have nice, well-understood properties in terms of how spaces embedded in these objects intersect". In [47], several constructions for *strictly functional repair* are proposed using a projective geometry viewpoint for a limited set of parameters, that tolerate multiple node failures. The constructions in [47] use well known combinatorial objects to construct spaces while controlling how they intersect. A simple example to illustrate the point is a set of three non-concurrent lines in a plane, which gives an exact repair code with the parameters $n = 3, k = d = 2$.

Despite these works, a general construction of functional regenerating codes with low overheads, achieving all the points on the trade-off curve between storage capacity and repair bandwidth, has not been achieved. This chapter addresses this problem by proposing an intuitive and practical construction to achieve functional repair for almost all admissible parameters. Our construction uses the geometric view of a linear repair code, and frames the relationship between the local intersection properties of the subspaces and the global independence properties of the subspaces. In the conference publication [87], we presented a code construction achieving the MBR point and an interior point on the optimal storage-repair bandwidth trade-off for full node repair. In this chapter, we modify and improve that scheme, and extend it to achieve all the points on the trade-off curve for full node repair as well as partial repair.

3.1.4 Our contributions

1. We derive the optimal storage-repair bandwidth trade-off for the broadcast repair of multiple partial failures, and discuss the advantage of repairing multiple

failed nodes simultaneously while leveraging unerased data remaining in the failed nodes for repair.

2. We derive invariant conditions that are sufficient to be satisfied for a functional repair code to be feasible. These conditions render structure to the random linear coding scheme, and provide an intuitive insight into the structure of a feasible functional repair code.

3. We present explicit codes that are optimal for functional repair of multiple nodes, with a probability close to one, for a large number of feasible parameters, in particular, for values of k which are divisible by r.

4. The proposed scheme employs an optimal (linear) subpacketization, has low computational complexity during the repair process, and has low overheads unlike random linear coding.

The chapter is organized as follows. The system model is introduced in Section 3.2. The storage-bandwidth trade-off for partial repair is derived in Section 3.3. Code constructions for multiple node repair and partial repair are presented in Section 3.4. Results and discussion are provided in Section 3.5. We close the chapter by highlighting our main results in Section 3.6.

3.2 System Model

Consider a wireless caching system where n nodes, each with storage capacity α bits, store a file of size M bits. We index these storage nodes by the set $\{1, \ldots, n\}$. The nodes are fully connected by a wireless broadcast medium and use orthogonal channels for data transmission. The file is divided into P data packets, and the number of data packets stored in each node, which is referred to as the *subpacketization* of an array code, is defined as $S = \frac{\alpha P}{M}$. The nodes store the file such that a user, modeled as a *data collector*, can reconstruct the file by obtaining the contents of any k nodes. This is called the *reconstruction property*.

We consider a scenario in which a portion of the stored bits in the storage nodes is subject to being lost. We refer to these nodes as the *faulty nodes* and to the nodes that do not experience any losses as the *complete nodes*. We assume that the repair occurs in rounds, where a repair round gets initiated when r nodes experience partial failures of $\alpha - \alpha_1$ bits, where $\alpha_1 \triangleq \rho\alpha, \rho \in [0, 1]$. Thus, a single repair round repairs r faulty nodes. There is no loss during a repair round. During a repair round, the lost bits in the faulty nodes are repaired with the help of transmitted bits from d complete nodes, $k \leq d \leq n - r$, called *helper nodes*, and the remaining bits that have not been lost in each of the faulty nodes. In general, the repair is functional, i.e., the repaired portion may not be the same as the original portion, but the repaired nodes satisfy the reconstruction property. See the table below for a list of the parameters associated with a regenerating code.

n	Number of storage nodes
k	Minimum number of nodes required for file reconstruction
r	Number of repaired nodes (newcomers) in each repair round
d	Number of helper nodes
M	File size in number of bits
α	Number of stored bits per node
α_1	Number of non-corrupted bits in a faulty node
ρ	Fraction of non-corrupted bits in a faulty node, i.e., $\alpha_1 = \rho\alpha$
β	Number of transmitted bits per helper node
γ	Total repair bandwidth, i.e, $\gamma = d\beta$

3.2.1 Information flow graph

The repair dynamics of the network can be represented by an information flow graph that evolves in time. See Fig. 3.2 for an illustration. It is a directed acyclic graph consisting of six types of nodes: a single source node S, storage nodes $x^i_{in}, x^i_{mid}, x^i_{out}$, auxiliary nodes h_i, and a data collector DC. Each complete storage node $x^i, i \in [n]$, is represented by two vertices: an input vertex x^i_{in} and an output vertex x^i_{out} that are

connected by a directed edge $x^i_{in} \to x^i_{out}$ with capacity α. A faulty node is represented by four vertices: an input vertex x^i_{in}, an intermediate vertex x^i_{mid} that is connected to x^i_{in} by a directed edge $x^i_{in} \to x^i_{mid}$ of capacity α, an output vertex x^i_{out} that is connected to x^i_{mid} by a directed edge $x^i_{mid} \to x^i_{out}$ of capacity α_1, and a failed vertex x^i_f that is connected to x^i_{mid} by a directed edge $x^i_{mid} \to x^i_f$ of capacity $\alpha - \alpha_1$. The failed vertex represents the corrupted portion of the data in the storage node.

Each vertex in the graph at any given time has two modes, active or inactive, depending on its availability. At the initial time, the source node S is active and it transmits data to n storage nodes such that the DC can retrieve the file from any k nodes. This is modeled by adding an edge from S to all input vertices of the storage nodes, $S \to x^i_{in}, i \in [n]$, with capacity ∞[1]. From this point onwards, the source node becomes inactive, and the storage nodes become active.

When r nodes experience partial failure of $\alpha - \alpha_1$ bits each, in the s-th round, the repair process is triggered and r newcomers join the system. Note that a newcomer represents the corresponding node being repaired. A newcomer x^i where $i = sn + j, j \in [n]$, represents the node x^j after the s-th round. The lost data is regenerated at the newcomers by receiving functions of the stored data from d helper nodes. d helper nodes are connected to the corresponding auxiliary nodes, denoted by h^i, with a directed edge $x^i_{out} \to h^i$ of capacity β, which denotes the number of bits broadcasted by x^i. Each auxiliary node h^i is connected with infinite capacity links to all the newcomers. This represents the broadcast nature of the transmission medium.

Definition 3.1. The repair bandwidth $\gamma = d\beta$ is defined as the total number of bits the helper nodes broadcast in a repair round.

We model a newcomer with two vertices x^i_{in} and x^i_{out} and a directed edge $x^i_{in} \to x^i_{out}$ with capacity α. The newcomer $x^i, i = n + (s-1)r + j$, uses the α_1 bits from the corresponding node being repaired. This is captured in the flow graph by edges with capacity $\alpha_1, x^i_{mid} \to x^i_{out}$, followed by the edges with infinite capacity between the output vertices of the node being repaired and the newcomers.

[1]Note that adding an edge with capacity ∞ means that all the information in the node sending the data is available in the input vertices of the nodes receiving the data.

A data collector corresponds to a request to reconstruct the file. Data collectors connect to any subset of k active nodes and retrieve all the stored data in these nodes, represented with edges with infinite capacity from the active nodes to the DC.

A cut-set in the information flow graph is a subset of edges such that there is no path from the source node S to the data collector DC that does not go through any of the edges in the cut-set. A cut partitions the graph into two disjoint sets of vertices, denoted by the pair $(U, \bar{U})$, where U is the set of vertices behind the cut, and $\bar{U}$ is the remaining vertices after the cut. We define the capacity of a cut-set as the sum of its edge capacities, and the *min-cut* of a graph as the cut-set with the minimum capacity among all the cut-sets.

Proposition 3.1. *[15] Consider any given finite information flow graph $\mathcal{G}$, with a finite set of data collectors. If the min-cut separating the source from each data collector is larger than or equal to the file size M, then there exists a linear network code defined over a sufficiently large finite field $\mathbb{F}$ (whose size depends on the graph size) such that all data collectors can reconstruct the original file. Further, randomized network coding guarantees that all collectors can reconstruct the file with probability that can be driven arbitrarily close to* 1 *by increasing the field size.*

Following Proposition 3.1, for the information flow graph construction described above, we find the minimum cut over all possible failure combinations. We enumerate cuts as $\chi_1, \chi_2, \ldots$ (see Fig. 3.2). In Section 3.3, we demonstrate how to find the min-cut for a specific example, and finally in the proof for Theorem 3.2 in Section 3.3, we describe the process for finding the min-cut for a general information flow graph.

3.3 Storage-Bandwidth Trade-off for Partial Repair

Consider the scenario illustrated in Fig. 3.2, where $n = 4, k = 2, d = 2$ and $r = 2$. The capacity of cut χ_1 is $2\alpha_1 + 2\beta$, while the capacity of cut χ_2 is 2α. Then the min-cut is $\min\{2\alpha_1 + 2\beta, 2\alpha\}$. From Proposition 3.1, to ensure that the file can be reconstructed by the data collector, $\min\{2\alpha_1 + 2\beta, 2\alpha\} \geq M$.

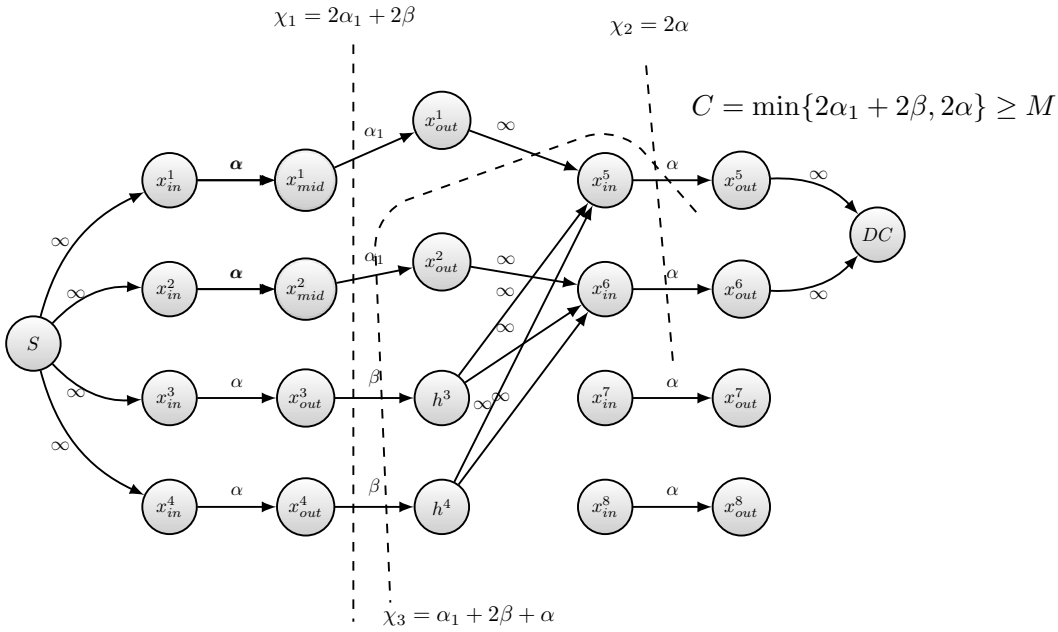


FIGURE 3.2: Information flow graph $\mathcal{G}$ with $n = 4, k = 2, r = 2$, one repair round and cuts χ_1, χ_2 and χ_3.

For each set of parameters $(n, k, d, \gamma, \alpha, r, \rho)$, there is a family of information flow graphs, each of which corresponds to a particular evolution of node failures/repairs. We denote this family of directed acyclic graphs by $\mathcal{G}(n, k, d, \gamma, \alpha, r, \rho)$. An $(n, k, d, \gamma, \alpha, r, \rho)$ tuple is feasible, if a code with storage α and repair bandwidth γ exists.

Theorem 3.2. *For any $\alpha \geq \alpha^*(n, k, d, \gamma, r, \rho)$, the points $(n, k, d, \gamma, \alpha, r, \rho)$ are feasible, and linear network codes suffice to achieve them. It is information theoretically impossible to achieve points with $\alpha < \alpha^*(n, k, d, \gamma, r, \rho)$. If r divides k, the threshold function $\alpha^*(n, k, d, \gamma, r, \rho)$ is given by:*

$$\alpha^*(n, k, d, \gamma, r, \rho) = \begin{cases} \frac{M}{k} & \gamma \in [f(0), \infty) \\ \frac{M - g(i)\gamma}{k - ir(1-\rho)} & \gamma \in [f(i), f(i-1)] \end{cases} \tag{3.1}$$

where, for $i = 1, 2, \ldots, \frac{k}{r} - 1$,

$$f(i) \triangleq \frac{2Md(1-\rho)}{(2k - ir(1-\rho))(i+1) + \frac{2k}{r}(d-k)}, \tag{3.2}$$

$$g(i) \triangleq (2d - 2k + r + ir)\frac{ir}{2d}. \tag{3.3}$$

Proof. Consider an information flow graph G that enumerates all possible failure/repair patterns and all possible data collectors when the number of failures/repairs is bounded by r. We analyze the connectivity in the information flow graph to find the minimum repair bandwidth. Initially, the source delivers α bits each to n nodes, which then become active while the source node becomes inactive. When r nodes lose part of their data, a repair round is triggered in which they connect to d surviving nodes and receive β bits from each of them. Using the received messages, and the un-erased content in their local memories, the r nodes recover their lost content. The active nodes, before the s-th repair round is triggered, are labelled as $\mathcal{R}_s^{act} \triangleq \{(s-1)n+1, \ldots, sn\}$. Consider that in the s-th repair round, the nodes $\mathcal{R}_s^{f} \triangleq \{(s-1)n+(s-1)r+1, \ldots, (s-1)n+sr\}$ have to be repaired. Denote the set of labels of the newcomers in the s-th round by $\mathcal{R}_s^{new} \triangleq \{sn+(s-1)r+1, \ldots, sn+sr\}$, which represent the repaired nodes. The complete nodes are copied into the next round and labeled as $\mathcal{R}_s^{comp} \triangleq \{\{sn+i\} : i \in [n] \setminus \mathcal{R}_s^{new}\}$. The newcomers and the copied complete nodes together form the set of active nodes for the next repair round, i.e., $\mathcal{R}_{s+1}^{act} = \mathcal{R}_s^{new} \cup \mathcal{R}_s^{comp}$, while the nodes from all the previous rounds become inactive.

First we show that any data collector DC that connects to the "out-nodes" of any k active nodes must satisfy

$$C = \text{mincut}(S, DC) \geq \sum_{s=1}^{k/r} \min\{(r\alpha_1 + (d - r(s-1))\beta, r\alpha)\}. \tag{3.4}$$

First, we show that there exists an information flow graph G' where the bound in Eq. (3.4) is matched with equality. Consider that after the storage nodes have gone through $s-1$ repair rounds, a data collector connects to the nodes with indices $\mathcal{R}_s^{new}$. Consider a cut $(U, \bar{U})$ between S and DC, which separates the graph into the disjoint sets of nodes U and $\bar{U}$, constructed as follows. For the s-th repair round, if $r\alpha \leq r\alpha_1+(d-(s-1)r)\beta$, then we include the nodes $x_{in}^{\mathcal{R}_s^{new}}$ in U, and $x_{out}^{\mathcal{R}_s^{new}}$ in $\bar{U}$, similarly to the cut χ_2 in Fig. 3.2; otherwise, we include $x_{in}^{\mathcal{R}_s^{new}}, x_{out}^{\mathcal{R}_s^{new}}, x_{out}^{\mathcal{R}_s^{f}}$, and all auxiliary nodes in $\bar{U}$, while the nodes $x_{mid}^{\mathcal{R}_s^{f}}$ and $x_{in}^{\mathcal{R}_s^{act}}$ are included in U, similarly to the cut χ_1 in Fig. 3.2. We argue that the capacity of the cut $(U, \bar{U})$ meets that of Eq. (3.4) with equality.

Second, we argue that any information flow graph has at least the minimum cut

capacity of Eq. (3.4). We note that there is a topological order of the nodes in an information flow graph by which any node ν_i having incoming edges only from nodes in $\bar{U}$, also belongs to $\bar{U}$, and an edge from ν_i to ν_j implies $i < j$. The min-cut in a repair round can be of only two types. In a "type-1" cut, all helper nodes not yet included in $\bar{U}$, and the out-vertices of the faulty nodes, are included in $\bar{U}$, as illustrated by the cut χ_1 in Fig. 3.2. A type-1 cut includes both the in-vertices and the out-vertices of the newcomer nodes in $\bar{U}$. In a "type-2" cut, all helper nodes and the in-vertices of the newcomers are included in U, while the out-vertices of the newcomers are included in $\bar{U}$, as illustrated by the cut χ_2 in Fig. 3.2. Other cuts that include the in-vertices of a subset of the newcomers in U, while including the in-vertices of the remaining newcomers in U, always have a capacity larger than type-1 cuts. We illustrate this using Fig. 3.2: suppose cut χ_3 passes through the edge $x^5_{in} \to x^5_{out}$, and the edges $x^2_{mid} \to x^2_{out}, x^3_{out} \to h^3, x^4_{out} \to h^4$. The capacity of cut χ_3 is given by $\alpha_1 + 2\beta + \alpha$, which is always greater than the capacity of the cut χ_1, since $\alpha_1 + 2\beta + \alpha > 2\alpha + 2\beta$. Therefore, we see that the minimum cut is always either type-1 or type-2 in any particular repair round, and the minimum cut thus obtained achieves the cut capacity given by Eq. (3.4).

The expression for the min-cut capacity is derived in the following way. The contribution of the first repair round to the minimum cut capacity is given by $\min\{r\alpha_1 + d\beta, r\alpha\}$, where the first term denotes the capacity contribution from a type-1 cut, and the second term denotes the contribution from a type-2 cut. Consider the second repair round. The r nodes which are repaired in the first repair round already lie in $\bar{U}$, so edges originating from these nodes do not contribute to the min-cut capacity from the second round onwards. The min-cut capacity contribution by the second repair round is given by $\min\{r\alpha_1 + (d-r)\beta, r\alpha\}$. We follow this procedure of passing the min-cut through each repair round with a type-1 or a type-2 cut until the DC lies in $\bar{U}$, which happens when the k nodes to which the DC is connected to lie in $\bar{U}$. When we sum the contributions from each repair round to the min-cut capacity, we obtain Eq. (3.4).

From Proposition 3.1, the min-cut capacity must be greater than the file size to ensure that the data collector is able to reconstruct the file from any k nodes. Therefore,

the following must be satisfied for guaranteed file reconstruction:

$$\sum_{s=0}^{k/r} \min\{(r\alpha_1 + (d - r(s-1))\beta, r\alpha)\} \geq M. \tag{3.5}$$

We are interested in characterizing the achievable trade-offs between the storage α and the repair bandwidth $d\beta$ when given (n, k, ρ). If $r\alpha \leq r\alpha_1 + (d-k+r)\beta$, then the min-cut is type-2 in each of the k/r repair rounds; if $r\alpha_1+(d-k+r)\beta \leq r\alpha \leq r\alpha_1+(d-k+2r)\beta$, then the min-cut is type-2 for the first $k/r-1$ repair rounds, but type-1 in the k/r-th repair round. In general, if $r\alpha_1+(d-rs)\beta \leq r\alpha \leq r\alpha_1+(d-r(s-1))\beta, s \in [k/r]$, then the min-cut is type-2 for the first s repair rounds, and type-1 for the remaining $k/r - s$ repair rounds.

Let $b_s \triangleq \frac{\frac{d-k}{r}+s+1}{1-\rho}\beta, s = [0 : k/r - 1]$. The capacity of the min-cut is a piecewise-linear function of α given by

$$C(\alpha) = \begin{cases} k\alpha & \alpha \in (0, b_0] \\ (k-r)\alpha + (r\alpha_1 + (d-k+r)\beta) & \alpha \in (b_0, b_1] \\ \vdots & \\ r\alpha + \sum_{i=1}^{k/r-1} (r\alpha_1 + (d-k+ir)\beta) & \alpha \in (b_{k/r-2}, b_{k/r-1}] \\ \sum_{i=1}^{k/r} (r\alpha_1 + (d-k+ir)\beta) & \alpha \in (b_{k/r-1}, \infty] \end{cases} \tag{3.6}$$

$$= \begin{cases} k\alpha & \alpha \in (0, b_0] \\ (k - ir(1-\rho))\alpha + (1-\rho)\sum_{j=0}^{i-1} rb_j & \alpha \in (b_{i-1}, b_i], i = 1, 2, \ldots, k/r - 1 \\ k\rho\alpha + (1-\rho)\sum_{j=0}^{k/r-1} rb_j & \alpha \in (b_{k/r-1}, \infty] \end{cases} \tag{3.7}$$

Note that $C(\alpha)$ is a strictly increasing function. To find the minimum α for a given repair bandwidth $\gamma = d\beta$ such that $C(\alpha) \geq M$, we let $\alpha^* = C^{-1}(M)$ to obtain

$$\alpha^* = \begin{cases} \frac{M}{k} & M \in (0, kb_0] \\ \frac{M - g(i)\gamma}{k - ir(1-\rho)} & M \in \Big[(k - ir(1-\rho))b_{i-1} + (1-\rho)\sum_{j=0}^{i-1} rb_j, \\ & \qquad (k - ir(1-\rho))b_i + (1-\rho)\sum_{j=0}^{i-1} rb_j\Big] \end{cases} \tag{3.8}$$

$$= \begin{cases} \frac{M}{k} & \gamma \in [f(0), \infty) \\ \frac{M - g(i)\gamma}{k - ir(1-\rho)} & \gamma \in [f(i), f(i-1)] \end{cases} \tag{3.9}$$

□

Corollary 3.1. *The minimum storage point is achieved by the pair*

$$(\alpha^*_{MSR}, \gamma^*_{MSR}) = \left(\frac{M}{k}, \frac{Mrd(1-\rho)}{k(d-k+r)} \right). \tag{3.10}$$

Corollary 3.2. *The minimum repair bandwidth point is achieved by the pair*

$$(\alpha^*_{MBR}, \gamma^*_{MBR}) = \left(\frac{2Md}{k(2d-(k-r)(1-\rho))}, \frac{2Mrd(1-\rho)}{k(2d-(k-r)(1-\rho))} \right). \tag{3.11}$$

Minimum-storage regenerating (MSR) and minimum-bandwidth regenerating (MBR) codes attain the points in Corollary 3.1 and Corollary 3.2, respectively.

Remark 3.1. *For $\rho = 0$ and $r = 1$, i.e., complete failure of exactly one node, the model is equivalent to that in [15], and the trade-off curve from Theorem 3.2 coincides with the trade-off curve in [15]. Similarly, for $\rho = 0$ and $r > 1$, i.e., multiple complete failures, the trade-off curve from Theorem 3.2 coincides with the trade-off curve in [38].*

Consider the parameters: $n = 4, k = 2, d = 3, r = 1, \alpha_1 = \frac{\alpha}{2}$. According to Eq. (6) in [37], the MSR point is achieved by the pair $(\alpha, \gamma) = (\frac{M}{2}, M)$. However, Corollary 3.1 suggests that a lower repair bandwidth with $\gamma = \frac{3M}{8}$ is achievable using linear network codes in $\mathbb{F}_q$, assuming q is large enough. Indeed, in Section 3.4, we show that this is achievable.

3.4 Code Construction

In this section, we present a framework for constructing explicit storage and repair schemes which achieve storage-repair bandwidth pairs that are very close to the points on the optimal trade-off curve. First, we provide a few preliminary concepts that are vital for the construction.

3.4.1 Subspace view

Consider that a node stores S linearly independent data packets $y_1, \ldots, y_S$ consisting of symbols in the finite field $\mathbb{F}_{q^l}$. For simplicity, assume that each data packet consists of exactly one symbol in $\mathbb{F}_{q^l}$. Finite field symbols in $\mathbb{F}_{q^l}$ can be viewed as $l-$dimensional vectors over $\mathbb{F}_q$, and therefore span a subspace. Hence, we say that the node stores a subspace of dimension S, denoted by $W_i = \text{span}\{y_i\}, i = 1, \ldots S$. For a set of nodes denoted by $\mathcal{A}$, the subspace stored by $\mathcal{A}$ is denoted by $W_{\mathcal{A}} = \sum_{i\in\mathcal{A}} W_i$. The sum of two vector spaces W_1 and W_2 is defined as $W_1 + W_2 = \{w_1 + w_2 : w_1 \in W_1, w_2 \in W_2\}$. Note that the sum of two vector spaces is not in general equal to their union. We use the notation $\dim\Big(.\Big)$ for the dimension of a vector space.

We list a few identities below associated with the dimension of vector subspaces. These identities are analogous to the information-theoretic identities. The analogy with information-theoretic identities may be further contextualized with the work in [88], by considering the dimension of a vector space as an information measure on it. To be precise, consider a random process $\tilde{W}^n$ of length n associated with a vector space W, that samples vectors from W with a uniform distribution at each time instant. Since there are at most $\dim(W)$ linearly independent vectors in W, the dimension of W serves as an information measure for the random process $\tilde{W}^n$ associated with the vector space W. Therefore, as described in [88], identities similar to those for Shannon information measures hold for the dimension of vector spaces.

Identities associated with the dimension of vector spaces:

1. *Conditional intersection:* We define the "conditional intersection" of a set of vector spaces $W_i, i = 1, \ldots, t$, conditioned on a vector space W_0, as the largest subspace in the intersection of the vector spaces $W_i + W_0, i = 1, \ldots, t$, after excluding the non-zero vectors belonging to the vector space W_0. Thus, we write the dimension of the conditional intersection as follows:

$$\dim\left(\bigcap_{i=1}^{t} W_i \Big| W_0\right) \triangleq \dim\left(\bigcap_{i=1}^{t} (W_i + W_0) \backslash W_0\right), \tag{3.12}$$

where $W_i \setminus W_j$, for two vector spaces W_i and W_j, denotes the largest subspace of W_i remaining after removing the non-zero elements belonging to $W_i \cap W_j$ from W_i. For $t = 2$, we note that the above identity becomes

$$\dim\left(W_1 \cap W_2 \middle| W_0\right) = \dim\left((W_1 + W_0) \cap (W_2 + W_0) \backslash W_0\right) \tag{3.13}$$

$$= \dim\left(W_1 \cap (W_2 + W_0) \backslash W_0\right), \tag{3.14}$$

where Eq. (3.14) follows due to the following reasoning: consider a vector w belonging to the vector space $(W_1 + W_0) \cap (W_2 + W_0)$ given by $w = w_1 + w_0^{(1)} = w_2 + w_0^{(2)}$, where $w_1 \in W_1, w_2 \in W_2$, and $w_0^{(1)}, w_0^{(2)} \in W_0$. Then, we also have $w' \in W_1 \cap (W_2 + W_0)$ given by $w' = w_1 = w_2 + (w_0^{(2)} - w_0^{(1)})$, where $w_0^{(2)} - w_0^{(1)} \in W_0$. Hence, the dimensions of the vector spaces $(W_1 + W_0) \cap (W_2 + W_0)$ and $W_1 \cap (W_2 + W_0)$ are equal. We can also deduce the following *chain rule* from Eq. (3.14):

$$\dim\left(W_1 \cap (W_2 + W_3)\right) = \dim\left(W_1 \cap W_2\right) + \dim\left(W_1 \cap (W_3 + W_2) \setminus W_2\right) \tag{3.15}$$

$$= \dim\left(W_1 \cap W_2\right) + \dim\left(W_1 \cap W_3 \middle| W_2\right) \tag{3.16}$$

2. $\dim\left(W_1 + W_2\right) = \dim\left(W_1\right) + \dim\left(W_2\right) - \dim\left(W_1 \cap W_2\right)$.
3. $\dim\left(W_1 \cap W_2\right) = \dim\left(W_1\right) - \dim\left(W_1 \setminus W_2\right)$.

3.4.2 Linearized polynomials

An important component in our construction is the linearized polynomial and its special properties. A linearized polynomial

$$f(x) = \sum_{i=1}^{P} a_i x^{q^{i-1}}, \quad a_i \in \mathbb{F}_{q^l} \tag{3.17}$$

can be uniquely identified from evaluations at any P points $x = \theta_i \in \mathbb{F}_{q^l}, i = 1, 2, \ldots, P$, that are linearly independent over $\mathbb{F}_q$. The polynomial interpolation problem (that is,

to determine the coefficients of $f(x)$ from the evaluations) can be written as

$$\mathbf{Qa} = \mathbf{y}, \tag{3.18}$$

where $\mathbf{Q}$ is the Moore matrix corresponding to the evaluation points ([89], chapter 1.3), $\mathbf{a} = (a_1, \ldots, a_P)^T$, and $\mathbf{y} = (f(\theta_1), \ldots, f(\theta_P))^T$. For linearly independent evaluation points $\theta_i, i = 1, \ldots, P$, $\mathbf{Q}$ is invertible, thus proving the existence of a unique solution for Equation (3.18).

Another relevant property of linearized polynomials is that they satisfy the following condition

$$f(ax + by) = af(x) + bf(y), \qquad a, b \in \mathbb{F}_q, \quad x, y \in \mathbb{F}_{q^l}. \tag{3.19}$$

In other words, given a set of points on a linearized polynomial, any linear combination over $\mathbb{F}_q$ of the points also lies on the polynomial.

3.4.3 General code construction for any point on the trade-off curve with $\rho = 0$ (full-node repair)

Substituting $i = \frac{k}{r} - \bar{j}, \bar{j} \in [\frac{k}{r}]$ in Theorem 3.2, we obtain the general expression for any point on the optimal storage-repair bandwidth trade-off as

$$(\alpha^*, \gamma^*) = \frac{M}{P^*}\Big(d - (\bar{j} - 1)r, rd\Big), \qquad \bar{j} \in \left[\frac{k}{r}\right], \tag{3.20}$$

where $P^* = k/2\Big(2\left(d - (\bar{j} - 1)r\right) - (k - r)\Big) + r\Big((\bar{j} - 1)k - \frac{\bar{j}(\bar{j}-1)}{2}r\Big)$. By considering $\frac{M}{P^*}$ as the size in bits of one data packet stored in a node, an optimal scheme stores $d-(\bar{j}-1)r$ data packets in a node, and has d helper nodes broadcasting rd data packets for the repair of r nodes in a repair round. Conversely, a scheme that stores $d-(\bar{j}-1)r$ data packets in a node, and has d helper nodes broadcasting rd data packets for the repair of r nodes in a repair round, is optimal if the size of each data packet is $\frac{M}{P^*}$. Note that the points on the trade-off curve are parametrized by $\bar{j}$, and are obtained by varying the values of $\bar{j}$.

3.4.3.1 Conditions for an optimal scheme

The following conditions **L1**,**L2**, and **L3**, are sufficient for an optimal repair scheme, and are described as follows:

L1: For any set of nodes $\mathcal{A}$ such that $|\mathcal{A}| \leq \bar{j}r$, the following holds : $\dim\Big(\sum_{i\in\mathcal{A}} W_i\Big) = \sum_{i\in\mathcal{A}} \dim\Big(W_i\Big)$. This further implies that $\dim\Big(W_{\mathcal{A}_1} \cap W_{\mathcal{A}_2}\Big) = 0$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are disjoint partitions of $\mathcal{A}$.

L2: Given a node A, and a set of nodes denoted by $\mathcal{B}$ such that $|\mathcal{B}| \leq d - (\bar{j}-1)r$. Partition $\mathcal{B}$ into two disjoint non-empty sets $\mathcal{B}_1$ and $\mathcal{B}_2$. Then the following holds:

$$\dim\Big(W_A \cap (W_{\mathcal{B}_1} + W_{\mathcal{B}_2})\Big) = \dim\Big(W_A \cap W_{\mathcal{B}_1}\Big) + \dim\Big(W_A \cap W_{\mathcal{B}_2}\Big). \quad (3.21)$$

This is equivalent to the following condition: Let $S_A^{\mathcal{B}_i}, \mathcal{B}_i \subset \mathcal{B}, i = 1, 2$, be the subspace broadcasted by node A to repair nodes in $\mathcal{B}_1$ and $\mathcal{B}_2$. Then, the following must hold:

$$\dim(S_A^{\mathcal{B}_1} \cap S_A^{\mathcal{B}_2}) = 0. \quad (3.22)$$

L3: Given a node A, and disjoint sets of r nodes denoted by $\mathcal{R}_1, \ldots, \mathcal{R}_{\bar{j}}$, the following holds:

$$\dim\left(W_A \cap W_{\mathcal{R}_{\bar{j}}} \Big| \sum_{i=1}^{\bar{j}-1} W_{\mathcal{R}_i}\right) \leq r. \quad (3.23)$$

3.4.3.2 Reconstruction

Suppose a DC accesses the nodes $1, \ldots, k$, denoted by $\mathcal{A}_{dc}$. For correct reconstruction, the data available in the k nodes should span the vector space spanned by the P packets of the file. Therefore, the necessary and sufficient condition for successful reconstruction is $\dim(W_{\mathcal{A}_{dc}}) \geq P$. In this section, we derive the dimension of the subspace stored by k nodes, assuming that the properties **L1**, **L2** and **L3** are satisfied by the storage nodes. First we propose the following lemma.

Lemma 3.1. *Assume that* **L1**, **L2** *and* **L3** *are satisfied. Given a node A, and a set of $v \leq d$ nodes denoted by $\mathcal{B}$, partition $\mathcal{B}$ into sets of r nodes denoted by $\mathcal{R}_1, \ldots, \mathcal{R}_{\lfloor v/r \rfloor}$, and denote the remaining set of nodes by $\mathcal{R}'$. Then,*

$$\dim\left(W_A \cap W_{\mathcal{B}}\right) = \sum_{s \geq \bar{j}}^{v/r} \dim\left(W_A \cap W_{\mathcal{R}_s} \Big| \sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\right) \tag{3.24}$$

Proof.

$$\dim\left(W_A \cap W_{\mathcal{B}}\right) = \dim\left(W_A \cap \sum_{i=1}^{v} W_i\right) \tag{3.25}$$

$$= \dim\left(W_A \cap \left(\sum_{t=1}^{v/r} W_{\mathcal{R}_t} + W_{\mathcal{R}'}\right)\right) \tag{3.26}$$

$$= \cancel{\dim\left(W_A \cap W_{\mathcal{R}_1}\right)} + \cancel{\dim\left(W_A \cap W_{\mathcal{R}_2} \middle| W_{\mathcal{R}_1}\right)} + \cdots + \cancel{\dim\left(W_A \cap W_{\mathcal{R}_{\bar{j}-1}} \middle| \sum_{t=1}^{\bar{j}-2} W_{\mathcal{R}_t}\right)}$$
$$+ \dim\left(W_A \cap W_{\mathcal{B}\setminus(\mathcal{R}_1,\ldots,\mathcal{R}_{\bar{j}-1})} \Big| \sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\right) \tag{3.27}$$

$$\overset{a}{=} \dim\left(W_A \cap W_{\mathcal{B}\setminus(\mathcal{R}_1,\ldots,\mathcal{R}_{\bar{j}-1})} \Big| \sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\right) \tag{3.28}$$

$$\overset{b}{=} \dim\left(W_A \cap W_{\mathcal{R}_{\bar{j}}} \Big| \sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\right) + \dim\left(W_A \cap W_{\mathcal{B}\setminus(\mathcal{R}_1,\ldots,\mathcal{R}_{\bar{j}})} \Big| \sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\right) \tag{3.29}$$

where step a is due to the property **L1**, and step b is due to **L2**. Using Eq. (3.29) recursively, we obtain

$$\dim\left(W_A \cap W_{\mathcal{B}}\right) = \sum_{s \geq \bar{j}}^{v/r} \dim\left(W_A \cap W_{\mathcal{R}_s} \Big| \sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\right) + \dim\left(W_A \cap W_{\mathcal{R}'} \Big| \sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\right) \tag{3.30}$$

$$\overset{c}{=} \sum_{s \geq \bar{j}}^{v/r} \dim\left(W_A \cap W_{\mathcal{R}_s} \Big| \sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\right) \tag{3.31}$$

where c follows from **L1**. □

Theorem 3.3. *If a DC accesses k nodes, denoted by $\mathcal{A}_{dc}$, $k \leq d$, then, assuming that* **L1**, **L2** *and* **L3** *are satisfied, we have*

$$\dim(W_{\mathcal{A}_{dc}}) \geq P^*, \tag{3.32}$$

thus satisfying the reconstruction property with optimal storage and repair bandwidth.

Proof.

$$\begin{aligned}
\dim\Big(\sum_{i=1}^{k} W_i\Big) &= \sum_{i=1}^{k}\left[\dim\Big(W_i\Big) - \dim\Big(W_i \cap \sum_{u=1}^{i-1} W_u\Big)\right] && (3.33)\\
&\overset{f}{=} \sum_{i=1}^{k}\dim\Big(W_i\Big) - \sum_{i=1}^{k}\sum_{s=\bar{j}}^{\lfloor (i-1)/r \rfloor}\dim\Big(W_i \cap W_{\mathcal{R}_s}\Big|\sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\Big) && (3.34)\\
&\geq k(d-(\bar{j}-1)r) - (r(r)+\cdots+r(k-\bar{j}r)) && (3.35)\\
&= \frac{k}{2}\left(2(d-(\bar{j}-1)r) - (k-r)\right) + r\left((\bar{j}-1)k - \frac{\bar{j}(\bar{j}-1)}{2}r\right) && (3.36)\\
&= P^* && (3.37)
\end{aligned}$$

where step f follows from Lemma 3.1. □

Therefore, a storage and repair scheme that divides the file into P data packets, and satisfies **L1**, **L2** and **L3**, achieves the optimal tradeoff between storage and repair bandwidth, by setting $P = P^*$. In the following section, we propose a scheme that satisfies **L1**, **L2** and **L3** with high probability.

3.4.3.3 Proposed code construction

A file of size M bits is divided into P data packets denoted by $m_1, \ldots, m_P$. For convenience and without loss of generality, we assume that each data packet consists of exactly one symbol in $\mathbb{F}_{q^l}$. Define the linearized polynomial

$$f(x) = \sum_{i=1}^{P} m_i x^{q^{i-1}}, \qquad m_i \in \mathbb{F}_{q^l}, \tag{3.38}$$

in a finite field $\mathbb{F}_{q^l}, l \geq P$. If a DC receives evaluations of the polynomial $f(x)$ on any P points in $\mathbb{F}_{q^l}$ that are linearly independent over $\mathbb{F}_q$, it can reconstruct $f(x)$ by interpolation, and thus reconstruct the file. For the rest of the section, we shall refer to evaluations of $f(x)$ on a set of linearly independent evaluation points as *linearly independent evaluations*. We propose a general scheme parameterized by $\bar{j}$ that achieves the points on the optimal storage-repair bandwidth trade-off with a high probability. Each node stores $d - (\bar{j} - 1)r$ linearly independent evaluations of $f(x)$.

We set the size of the finite field to q^l, where $l \geq (n-r)(d-(\bar{j}-1)r)$, and store $d-(\bar{j}-1)r$ linearly independent evaluations of $f(x)$ on the nodes $1, \ldots, n-r$. Subsequently, the contents of the remaining $n-r$ nodes are generated by the nodes $1, \ldots, d$ by using the repair scheme described in the next subsection, as if the d nodes are helper nodes repairing the nodes $n-r+1, \ldots, n$. This ensures that the conditions **L1**, **L2**, and **L3** are satisfied in the initial storage round.

3.4.3.4 Repair scheme

The following repair scheme satisfies the properties **L1**, **L2** and **L3**, after an arbitrary number of repair rounds, with high probability. During repair, the d helper nodes, denoted by the set $\mathcal{H}$, transmit r packets each to repair r newcomers, enumerated as $\{n_1, \ldots, n_r\} = \mathcal{N}$. Node $h_i \in \mathcal{H}$, for $i = 1, \ldots, d$, transmits r random linear combinations of packets in set $\mathcal{A}, |\mathcal{A}| = r + e, 0 \leq e \leq d - \bar{j}r$, where set $\mathcal{A}$ consists of $r+e$ packets sampled randomly from the packets stored in node h_i, and e is a parameter that can be set based on a hit and trial method. The packets transmitted by the helper nodes h_i in set $\mathcal{H}$, enumerated as $w_{h_i,1}, \ldots, w_{h_i,r}$, are received and arranged by each

newcomer in a matrix $\mathbf{Y}$ of dimensions $(d-(\bar{j}-1)r) \times \bar{j}r$ in the following manner:

$$\mathbf{Y}^T = \begin{bmatrix} w_{h_1,1} & w_{h_2,1} & \cdots & w_{h_{d-(\bar{j}-1)r},1} \\ \vdots & \ddots & \ddots & \vdots \\ w_{h_1,r} & w_{h_2,r} & \cdots & w_{h_{d-(\bar{j}-1)r},r} \\ w_{h_{r+1},1} & w_{h_{r+2},1} & \cdots & w_{h_{d-(\bar{j}-2)r},1} \\ \vdots & \ddots & \ddots & \vdots \\ w_{h_{r+1},r} & w_{h_{r+2},r} & \cdots & w_{h_{d-(\bar{j}-2)r},r} \\ w_{h_{2r+1},1} & w_{h_{2r+2},1} & \cdots & w_{h_{d-(\bar{j}-3)r},1} \\ \vdots & \ddots & \ddots & \vdots \\ w_{h_{(\bar{j}-1)r+1},r} & \cdots & \cdots & w_{h_d,r} \end{bmatrix}. \tag{3.39}$$

Note that all the r packets received from the d helper nodes are present in matrix $\mathbf{Y}$ with a certain symmetrical arrangement.

Now, indexing the rows of matrix $\mathbf{Y}^T$ from 0 to $\bar{j}r-1$, the following rotation operation is done on each row of $\mathbf{Y}^T$:

$$\text{Rotate}_{(g \bmod r)}\,(\text{row } g)\,, \qquad g = 0, \ldots, \bar{j}r-1, \tag{3.40}$$

where the function $Rotate_{\sigma}(v)$ does circular rotation of the vector v by σ positions. Thus, newcomer $n_i, i = 1, \ldots, r$, obtains matrix $\mathbf{Y}$ such that each row contains packets from $\bar{j}r$ distinct helper nodes. Newcomer n_i computes random linear combinations over $\mathbb{F}_q$ of the $\bar{j}r$ packets in each row of $\mathbf{Y}$, and stores the resultant packets in its memory.

In the following, we argue that the conditions **L1**, **L2** and **L3** are satisfied by the above repair scheme with a high probability.

L1: Consider a non-zero random linear combination v_{p+1} of a set V of p vectors $v_1, \ldots, v_p$. Then, the set of vectors $\{v_{p+1}\} \cup V', V' \subset V, |V'| = p-1$ forms a basis. Moreover, a set containing t non-zero random linear combinations of the p vectors in V, and any $p-t$ vectors in V, forms a basis with a very high probability if the field size is sufficiently large. Therefore, since each newcomer stores

a random linear combination of $\bar{j}r$ packets received from $\bar{j}r$ distinct nodes, the property **L1** is satisfied with a high probability which approaches 1 if q, the size of the base field, is sufficiently large.

L2: Helper h_i transmits r random linear combinations of $r + e$ linearly independent packets in each repair round to repair a group of r newcomers. Therefore, Eq. (3.22) is satisfied with a high probability, if e is chosen to be sufficiently large, and q is sufficiently large.

L3: This property holds true for the described scheme with a high probability, because each helper node delivers r linearly independent packets to the newcomers, and these are linearly combined with the packets from $\bar{j} - 1$ other helper nodes. Therefore, given the packets from the $\bar{j} - 1$ helper nodes, node A and the r nodes in the set $\mathcal{R}_1$ have only r packets in common with a high probability.

3.4.4 MSR point with $\rho = 0$

To achieve the MSR point (see Corollary 3.1), we set $\bar{j} = \frac{k}{r}$ in the code construction in Section 3.4.3. Thus the file is divided into $P = P^* = k(d - k + r)$ data packets. Each node stores $d - k + r$ linearly independent points on $f(x)$. $\mathbf{Y}$ is of dimensions $(d - k + r) \times k$, and thus each newcomer computes random linear combinations of the k packets in the rows of $\mathbf{Y}$. In this manner, an MSR code is constructed with optimal subpacketization $S = d - k + r$.

3.4.5 MBR point with $\rho = 0$

To achieve the MBR point (see Corollary 3.2), we set $\bar{j} = 1$ in the code construction in Section 3.4.3. The file is divided into $P = P^* = \frac{k}{2}(2d - k + r)$ data packets. Each node stores d linearly independent points on $f(x)$. $\mathbf{Y}$ is of dimensions $d \times r$, and each newcomer computes random linear combinations of the r packets in the rows of $\mathbf{Y}$.

3.4.6 Code construction for any point on the trade-off curve with $\rho > 0$ (partial repair)

In this section, we propose the sufficient conditions for an optimal partial repair scheme, and then propose an extension of the repair scheme from the previous section to achieve optimal partial node repair performance with high probability. Substituting $i = \frac{k}{r} - \bar{j}, \bar{j} \in [\frac{k}{r}]$ in Theorem 3.2, we obtain the general expression for any point on the optimal storage-repair bandwidth trade-off for partial repair as

$$(\alpha^*, \gamma^*) = \frac{M}{P^*}\Big(d - (\bar{j}-1)r, rd(1-\rho)\Big), \qquad \bar{j} \in \left[\frac{k}{r}\right], \tag{3.41}$$

where $P^* = \frac{k}{2}\left(2(d-(\bar{j}-1)r) - (1-\rho)(k-r)\right) + r(1-\rho)\left((\bar{j}-1)k - \frac{\bar{j}(\bar{j}-1)}{2}r\right)$. An optimal partial repair scheme divides a file into P^* data packets, and each node stores $(d-(\bar{j}-1)r)$ coded packets. Instead, we consider that, for $\xi \in \mathbb{N}$, such that $\rho\xi \in \mathbb{N}$, the file is divided into ξP^* data packets, and each node stores $(d-(\bar{j}-1)r)\xi$ coded packets. The storage capacity and the repair bandwidth achieved is optimal for any arbitrary ξ.

3.4.6.1 Conditions for an optimal scheme

The properties **L1** and **L2** remain the same as in Section 3.4.3. The property **L3** is described as follows:

L3: Given a node A, and $\bar{j}$ disjoint sets of r nodes denoted by $\mathcal{R}_1, \ldots, \mathcal{R}_{\bar{j}}$, the following property holds:

$$\dim\left(W_A \cap W_{\mathcal{R}_{\bar{j}}} \Big| \sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\right) \leq (1-\rho)r\xi. \tag{3.42}$$

3.4.6.2 Reconstruction

Given that **L1**, **L2** and **L3** are satisfied, the dimension of the space obtained by a DC accessing any k nodes is given by

$$\dim\Big(\sum_{i=1}^{k} W_i\Big) = \sum_{i=1}^{k}\left[\dim\Big(W_i\Big) - \dim\Big(W_i \cap \sum_{u=1}^{i-1} W_u\Big)\right] \tag{3.43}$$

$$= \sum_{i=1}^{k}\dim\Big(W_i\Big) - \sum_{i=1}^{k}\sum_{s=\bar{j}}^{\lfloor (i-1)/r \rfloor}\dim\Big(W_i \cap W_{\mathcal{R}_s}\Big|\sum_{t=1}^{\bar{j}-1} W_{\mathcal{R}_t}\Big) \tag{3.44}$$

$$\geq k\xi\Big(d-(\bar{j}-1)r\Big) - \Big((1-\rho)\xi r(r) + \cdots + (1-\rho)\xi r(k-\bar{j}r)\Big) \tag{3.45}$$

$$= k\xi\Big(d-(\bar{j}-1)r\Big) - \xi\Big((1-\rho)r(r) + \cdots + (1-\rho)r(k-r)\Big) + \xi(1-\rho)r\Big((k-(\bar{j}-1)r) + \cdots + (k-r)\Big) \tag{3.46}$$

$$= \frac{k\xi}{2}\left(2(d-(\bar{j}-1)r) - (1-\rho)(k-r)\right) + \xi r(1-\rho)\left((\bar{j}-1)k - \frac{\bar{j}(\bar{j}-1)}{2}r\right), \tag{3.47}$$

which is equal to P^*, thus proving optimality. Therefore, if we have a repair scheme for which **L1**, **L2** and **L3** are satisfied after an arbitrary number of repair rounds, the value of P can be set to P^*, thus achieving the optimal performance in terms of storage and repair bandwidth.

3.4.6.3 Proposed code construction

The general code construction for partial repair is an extension of that presented in Section 3.4.3. We consider that, for $\xi \in \mathbb{N}$, such that $\rho\xi \in \mathbb{N}$, the file is divided into ξP^* data packets, and each node stores $(d-(\bar{j}-1)r)\xi$ coded packets. Thus, when $(1-\rho)(d-(\bar{j}-1)r)\xi$, where $0 \leq \rho < 1$, packets are erased on each of the r faulty nodes, we have an integer number of erased packets, assuming that $\rho\xi \in \mathbb{N}$. The linearized polynomial $f(x)$ is constructed with these packets as coefficients, similarly to Eq. (3.38). Node $i, i = 1, \ldots, n$, stores $S = \Big(d-(\bar{j}-1)r\Big)\xi$ linearly independent evaluations of $f(x)$, enumerated as $w_{i,j}, j = 1, \ldots, (d-(\bar{j}-1)r)\xi$.

3.4.6.4 Repair scheme

We assume that $(1-\rho)(d-(\bar{j}-1)r)\xi$ packets are erased on each of the r faulty nodes. Consider that the indices of the r faulty nodes are denoted by the set $\mathcal{N}$, and the indices of the helper nodes are denoted by $\mathcal{H}$. During repair, the helper node $h \in \mathcal{H}$ transmits $(1-\rho)\xi r$ random linear combinations of a set of $(1-\rho)(r+e)\xi$ randomly sampled packets. The number of unerased packets on a faulty node $n_i \in \mathcal{N}, i = 1, \ldots, r$, is given by $\rho\xi\left(d-(\bar{j}-1)r\right)$. A set of $r\xi\rho$ packets are randomly sampled from the unerased packets in the faulty node n_i, which are added to the $(1-\rho)r\xi$ packets transmitted by a helper node, thus making $r\xi$ packets per helper node. Faulty node n_i arranges these packets to form a matrix $\mathbf{Y}$ of dimensions $(d-(\bar{j}-1)r)\xi \times \bar{j}r$, such that each row contains packets from $\bar{j}r$ distinct helper nodes. The conditions **L1**, **L2** and **L3** are satisfied with high probability, and the rest of the repair scheme proceeds similar to that in Section 3.4.3.4.

3.5 Results and Discussion

3.5.1 Optimal trade-off curve

In Fig. 3.3, we plot the storage vs. repair bandwidth trade-off for single-node repair [15], broadcast repair of multiple full node failures [38], and partial repair of multiple nodes. Fig. 3.3 illustrates that utilizing the un-erased portion of data on a failed node reduces the repair bandwidth significantly. The repair bandwidth is normalized by the number of failed nodes, and the fraction of data that is erased, in order to make a fair comparison with single node repair and full node repair respectively. We call it the *normalized repair bandwidth*. We observe that the MSR point does not improve over full node repair even with partial repair, that is, the un-erased data in a faulty node does not help in the repair of the erased packets in any meaningful way for the MSR point. However, the normalized repair bandwidth decreases significantly at all other points on the trade-off curve.

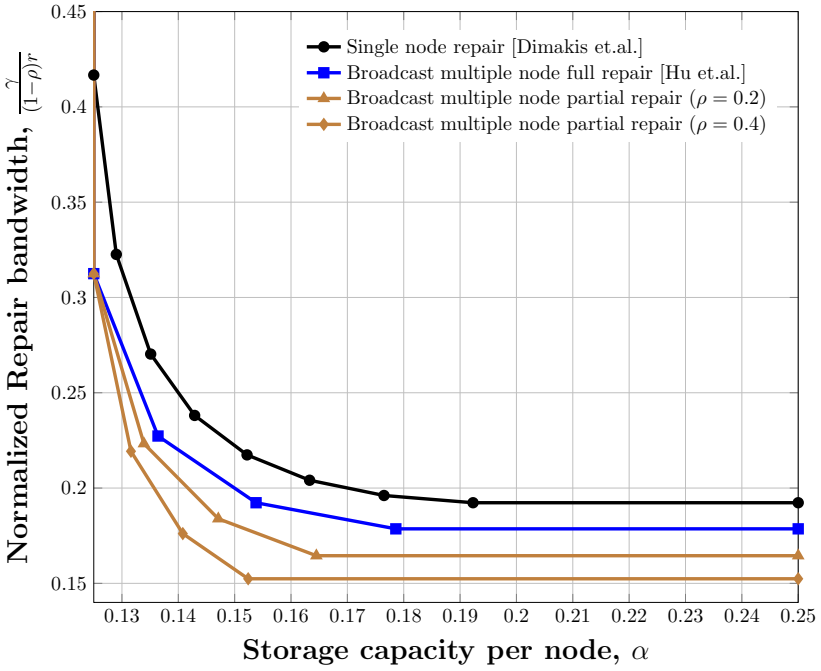


FIGURE 3.3: Trade-off curve between the repair bandwidth and storage, $M = 1, k = 8$ and $d = 10$ helper nodes. For single node failure $r = 1$ and for multiple node failures $r = 2$.

3.5.2 Performance of the code construction

3.5.2.1 Achievable storage-repair bandwidth pairs

The following table records the results of experiments with multiple parameter combinations, for different points on the corresponding trade-off curves for those parameters, using the proposed scheme for full-node repair. We run 100 repair rounds, where the indices of the failing nodes and the helper nodes are sampled randomly. We then compute the dimensions of the subspace obtained from sets of k nodes, sampled randomly in each of 50 trials, and record the minimum and the average dimension of the subspaces obtained from the sets of k nodes observed in the trials. We observe that $\dim(W_{\mathcal{A}_{dc}}) \geq P^*$ is satisfied almost surely for arbitrary parameter combinations if e and q are chosen appropriately.

n	k	d	r	$\bar{j}$	q	e	P^*	min	avg
27	15	17	5	1	29	0	180	191	191
				2	29	0	155	158	158
				3	257	2	105	105	105
24	16	16	4	1	29	1	160	170	170
				2	29	1	144	149	149
				3	29	1	112	114	114
				4	29	0	64	64	64
20	12	12	4	1	29	1	96	96	96
				2	29	1	80	84	84.98
				3	29	0	48	48	48
16	12	12	3	1	1021	3	90	91	91
				2	1021	3	81	83	83
				3	257	3	63	63	63
				4	257	0	36	36	36
16	8	11	2	1	29	1	64	65	74.1
				2	29	1	60	64	68.84
				3	29	1	52	54	55.56
				4	29	1	40	40	40
14	10	10	2	1	29	2	60	60	60
				2	29	1	56	57	57.96
				3	29	2	48	49	49
				4	29	2	36	36	36.92
				5	127	0	20	20	20
9	6	6	3	1	1021	3	27	27	27
				2	1021	0	18	18	18

3.5.2.2 Subpacketization

The proposed scheme achieves the optimal linear subpacketization for all points on the trade-off curve for full node repair. The subpacketization is given by $S = d-(\bar{j}-1)r$

for different points on the trade-off curve parameterized by $\bar{j}$. For partial repair, the subpacketization is given by $S = (d - (\bar{j} - 1)r)\xi, \xi \in \mathbb{N}$, which is also linear in n, k, d and r.

3.5.2.3 Input/Output cost

The input-output cost is defined as the number of symbols that need to be read by a helper node from its memory, which are then linearly combined and sent to the newcomers. For the repair of r newcomers, the optimal input-output cost is given by r, while the proposed scheme achieves an input-output cost of $r + e$, where $e = 0$ implies the optimal input-output cost. As recorded in the table above, small values of e are often sufficient to achieve optimal performance, implying a low input-output cost.

3.5.2.4 Computational complexity

The finite field operations are in the finite extension field $\mathbb{F}(q^l)$. For reconstruction of the file from any k nodes, the data collector interpolates a linearized polynomial of degree P in the finite field $\mathbb{F}_{q^l}$. The complexity of interpolation of a linearized polynomial is $\mathrm{O}(P^\epsilon)$ operations in $\mathbb{F}_q$, where ϵ is the matrix multiplication exponent [90]. One of the frequently used fast algorithms for matrix multiplication is known as the Strassen algorithm [91], which achieves the matrix multiplication exponent $\epsilon = 2.807$. Therefore, the fastest reconstruction complexity is $\mathrm{O}(P^{2.807})$ operations in $\mathbb{F}_q$.

The repair complexity, defined as the computational complexity of repair operations, is smaller than that of random linear coding, because only rows of $\bar{j}r < dr$ packets are multiplied with the parity matrix $\mathbf{M}_{\bar{j}r \times r}$ in the repair process, which makes the computations faster, unlike random linear coding where, in general, d random linear combinations of dr packets are computed during the repair process, requiring d^2r operations. Thus, the time complexity for random linear coding in each newcomer is $O(d^2r)$, while that for the proposed scheme is $O(dr)$, achieving a reduction in the repair complexity by a factor of d.

3.6 Conclusions

In this chapter, we studied large scale distributed storage systems and considered the problem of repair of partial failures of multiple nodes via broadcast transmissions over a wireless medium. We derive the optimal storage-repair bandwidth trade-off curve by constructing an information flow graph to represent the evolution of the system with time, and computing the minimum cut-set capacity in the information flow graph. It has been shown in previous literature that compared to single node repair, repairing multiple nodes at once and exploiting the broadcast nature of the medium reduces the repair bandwidth per failed node. We illustrate that the optimal repair bandwidth is reduced even further by utilizing the remaining content in the nodes that experience partial failure. We derive the invariant conditions related to the way the subspaces stored by different nodes intersect, that are sufficient for the existence of a feasible functional regenerating code, and that provide an intuitive insight into how functional regenerating codes may be constructed. We then present an explicit storage and repair framework for the functional repair of multiple node failures in a broadcast setting, achieving all the points on the trade-off curve, as illustrated in Fig. 3.3, with high probability. We also extend the framework for the case when there is only partial failure of multiple nodes. The proposed storage and repair framework achieves multiple desirable characteristics for regenerating codes. These characteristics include achieving the optimal storage-repair bandwidth trade-off with high probability, for *many feasible parameters* (n, k, d, r) not achieved in existing literature; achieving *all points on the trade-off curve*; optimal linear subpacketization with respect to the parameters; low input-output cost; and low computation complexity during repair.

Chapter 4

Secure Distributed Matrix Computations

4.1 Introduction

In the era of big data, performing computationally intensive operations locally on a single machine is infeasible, and clients often rely on powerful cloud servers for demanding computations. In the so-called *serverless computing paradigm*, clients can request computationally expensive tasks to be performed on massive datasets, potentially generated at multiple geographically distributed locations, using special purpose computing servers (eg., Amazon Web Services (AWS), Microsoft Azure, Google Cloud). While serverless computing provides significant flexibility and speed up, it also leads to growing data privacy concerns, as the corporations that provide computation services also provide many other digital services, and have access to unprecedented amounts of private user data. Therefore, algorithms that would allow users to benefit from powerful untrustworthy servers while keeping their data private are of significant interest. Our goal in this chapter is to design efficient secure distributed matrix computation (SDMC) algorithms, which keep data private from the potentially colluding computing servers as well as the entities requesting the computations.

We consider $\Gamma \geq 1$ data *sources*, represented as matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(\Gamma)}$ on an appropriate finite field. A *user* requests to compute a function of these matrices, $G(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(\Gamma)})$, with the help of N computing servers. All of the sources are connected to the servers, and all the servers are connected to each other, with orthogonal and reliable links. Similarly, computations carried out by the servers are conveyed to the user over orthogonal and reliable links. For a given number of N servers, our goal

will be to minimize the amount of data that must be uploaded from the sources to the servers, which we refer to as the *upload cost*. The upload cost often determines the financial cost of serverless computing, but minimizing it would also reduce the overall computing time as it limits the amount of computations that must be carried out by the servers, as well as the data delivery time from the sources to the servers, which may be prohibitive especially when the data sources are geographically distant from the servers. For example, the source nodes may be geographically distributed hospitals sharing medical data of patients, and the user may be a research institute or a pharmaceutical company making certain queries on the data. In addition to correct computation of the request, we also want to guarantee the privacy of the input data against the servers as well as the requesting user. We impose information theoretic perfect privacy guarantees such that any T colluding servers must not learn anything about the data sources, or the user must not learn anything about the data sources apart from the computation result. We assume that all the servers are honest and responsive, but curious, which means that they follow the prescribed protocol honestly, but any T of them may collude to try to deduce information about the input matrices. We will also consider the special setting of *computation with own data*, in which case the user wants to compute a function on its own data matrices using the available computing servers. In this case we drop the privacy requirement against the user, and the problem lends itself to further optimization.

We will first focus on the secure distributed matrix multiplication (SDMM) problem, which has received significant recent interest. Large scale matrix multiplication is a fundamental building block of matrix computations in many machine learning, optimization, and signal processing algorithms. It is also one of the most computationally intensive operations. Moreover, it can be easily distributed across multiple servers thanks to its inherently parallel structure. We will first consider the multiplication of $\Gamma = 2$ matrices, which will allow us to introduce the main ideas behind our design. We then extend our analysis to the multiplication of multiple matrices, as well as other fundamental operations which, when combined with matrix multiplication, allow computation of arbitrary polynomials of matrices.

4.1.1 Related Work

The cryptography community has extensively studied the problem of secure multi-party computation (MPC), also known as secure function evaluation, in which Alice and Bob, having inputs x and y, respectively, want to compute a function $f(x, y)$ jointly, without any of them learning anything about the other's input either from the communication, or from the result of the computation [50]. The SDMM problem is related to MPC yet different; the design has to ensure that no computing server learns anything about the original data, but we can decide which part of the data is revealed to each server and in what form. Fully homomorphic encryption (FHE) is a class of cryptographic schemes that allow computations on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on plaintext. These techniques rely on working over polynomial rings, and their security is based on the assumed (or proven) hardness of problems in ideal lattices [51, 52]. However, existing FHE schemes are slow and impractical. "Somewhat homomorphic encryption" (SHE) has been proposed as an alternative, which allows a limited number of homomorphic operations on ciphertexts. SHE is relatively faster, and ciphertext packing methods have been proposed for operations like secure inner products [51,53], and secure matrix multiplications [52], which generalizes Yasuda et al.'s packing method for inner products in [53]. Some works also propose methods for performing other matrix computations in an information theoretically secure manner, like Gaussian elimination, matrix inversion, comparison, equality test, or exponentiation [92, 93].

There is also a growing literature on distributed matrix multiplication, where a lot of effort has been put into speeding up computations, increasing reliability, and/or reducing communication overhead using coding and communication theoretic ideas [54–60]. The initial papers considered a slightly different context of speeding up parallel computations by introducing "computation redundancy" to mitigate the problem of straggling servers [54,55]. Straggling servers refer to slow/unresponsive servers due to which completion of the computation is delayed. A standard way of dealing with stragglers is to introduce "computation redundancy", that is, assigning extra computations to each

server. The common theme in the 'coded computation' literature is to treat stragglers as 'erasures' in communications, and apply ideas for coding against erasures, which allows reliable reconstruction of the desired result from an arbitrary set of successfully received symbols. Reference [56] uses polynomial codes to construct a scheme in which the computation is completed as long as any K out of N evaluations of a polynomial are received from the servers. To multiply two matrices $\mathbf{A}$ and $\mathbf{B}$ with the help of N servers, the polynomial code in [56] partitions $\mathbf{A}$ row-wise and $\mathbf{B}$ column-wise (row-by-column partitioning), and generates encoded matrices as evaluations of a polynomial with the blocks as the coefficients, similarly to Reed Solomon codes [61]. In coded computation against stragglers, the performance metric is the *recovery threshold*; that is, the minimum number of successful (non-delayed, non-faulty) servers required for completing the computation. A follow-up work [57] proposes a new polynomial coded computation scheme called MatDot, which achieves the optimal recovery threshold for column-wise partitioning of matrix $\mathbf{A}$ and row-wise partitioning of $\mathbf{B}$ (sum-of-outer-products method), which we shall refer to as the column-by-row partitioning henceforth in the chapter, at the expense of an increase in the communication cost. In [57], the authors also propose PolyDot codes that interpolate between the polynomial codes of [56] and MatDot codes. PolyDot codes are later improved to generalized PolyDot (GPD) codes in [58], which achieve the optimal recovery threshold for any arbitrary partitioning of the input matrices. Entangled polynomial codes, proposed in parallel in [59], also achieve the same performance as GPD codes. A bivariate polynomial code is introduced in [94] for straggler mitigation.

Subsequent papers, inspired by the works on straggler mitigation in distributed matrix multiplication, consider the SDMM problem from an information theoretic perspective. These papers aim for information theoretic security, independent of the computational capacities of the attackers, as opposed to cryptographic techniques. These works typically assume that the data to be used for the computation is available at the user, and hence the main concern is the privacy against the servers. The earlier papers on SDMM consider *download rate* as the performance metric, which is defined as the ratio of the number of bits required to represent the computation result to the total number of bits that the servers must transmit to the user. Reference [62] uses

the idea of polynomial codes from the literature on straggler mitigation to propose an SDMM scheme based on Shamir's secret sharing scheme [63], which is shown to achieve the optimal download rate for *one-sided* SDMM (where only one of the matrices is kept secure). An achievable scheme is also proposed for *two-sided* SDMM (both matrices are kept secure). Reference [64] introduces GASP codes, which improve the download rate for two-sided SDMM by aligning the degrees of the terms in the polynomial code so that the desired products appear as distinct terms.

In [65], new converse bounds on the optimal download rate for SDMM are obtained by showing that the capacity of a multi-message X-secure T-private information retrieval (MM-XSTPIR) problem ([66, 67]) provides an upper bound on the download rate of the SDMM problem. The optimal download rate of the MM-XSTPIR problem is shown to depend on the dimensions of the matrices $\mathbf{A}$ and $\mathbf{B}$. The scheme in [65] allows the joint retrieval of a batch of matrix products (*batch multiplication*), instead of multiplying two matrices using the *matrix partitioning* approach, resulting in a coding gain.

Reference [60] combines straggler mitigation and secure computation using the batch multiplication approach. Byzantine security is also considered, which refers to security against adversarial servers that may actively corrupt the results they send back to the user. Their scheme is based on Lagrange polynomials, and it achieves the optimal recovery threshold for any multi-linear function computation. Since the scheme in [60] is designed for batch computation of any function, we can adapt it for matrix multiplication using a matrix-partitioning based approach, and compare its performance with that of other matrix partitioning based schemes. This can be done by treating the partitions of the matrices to be multiplied as batches of data. Therefore, if $\mathbf{A}$ is partitioned column-wise and $\mathbf{B}$ is partitioned row-wise into K partitions each, the scheme in [60] has the recovery threshold of $2(K + T - 1) + S + 2A + 1$, where S and A are the numbers of stragglers and Byzantine adversaries, respectively. If $\mathbf{A}$ is partitioned row-wise and $\mathbf{B}$ is partitioned column-wise, into K and L partitions respectively, the recovery threshold is $2(KL + T - 1) + S + 2A + 1$, which is the same performance as that of the GASP codes in [64] for big T, which can be seen by setting $S = A = 0$. For arbitrary matrix partitions, the Lagrange coded scheme does

not perform as well as the secure generalized PolyDot (SGPD) codes introduced later in [68]. In [68], the trade-off between the download rate and the recovery threshold, first studied in [69] using the MatDot and PolyDot schemes for straggler mitigation, is extended to SDMM. The SGPD codes achieve the same recovery threshold as Lagrange codes for the sum-of-outer-products method.

In [95], the trade-off between the upload and download costs for SDMM is studied. While the *download cost* is simply the reciprocal of the download rate, the *upload cost* is defined as the ratio of the total number of bits that the user must send to the servers to the total size in bits of the data matrices. In [95], a secure cross subspace alignment (SCSA) scheme with adjustable upload cost (USCSA) is presented for SDMM. The tradeoff between the upload and download costs for SDMM schemes using only the row-by-column partitioning is studied in [95], but not the tradeoff for SDMM schemes using the column-by-row partitioning.

The work that is most related to ours is [70], which considers the setting in which the data is generated at distributed source nodes, and does not belong to the user requesting the computation. It extends the BGW (Ben-Or, Goldwasser and Widgerson) scheme from [71], which was first proposed in the context of secure MPC, for multiplication of matrices using a connected network of computing servers. The sources are assumed not to be connected with each other, while the servers are. The latter assumption is exploited to reduce the communication to the user by allowing the servers to cooperate securely. Thus, the servers share their results from the first round of computation among each other using Shamir's secret sharing scheme, and compute a linear combination of the received shares of the results. This inter-server cooperation allows a smaller number of servers to send these linear combinations to the user. This particular model also imposes privacy against the user, that is, the user cannot learn anything about the data beyond what it learns from the computation result. This constraint is not imposed in other papers, where the user is the source of the input data; and neither in [65], where distributed source nodes generate the data.

4.1.2 Main contributions

With respect to the rich literature on the topic that we have summarized above, our main novel contributions described in this chapter can be summarized as follows:

- We first introduce a novel polynomial coding scheme exploiting the properties of discrete Fourier transform, and show that it achieves a near optimal upload cost for SDMM of two matrices, while achieving the optimal upload cost for the special case of computation with own data, in which the user has access to the matrices used in the computations.
- The proposed scheme can be implemented in an efficient manner using the recently developed fast Fourier transform (FFT) algorithm on finite fields [72], and has negligible decoding complexity.
- We generalize the proposed scheme for SDMM of two matrices to introduce straggler mitigation.
- We extend the proposed scheme to multiply multiple matrices on distributed servers securely, while satisfying the security constraints for keeping the intermediate and final results secure against any T colluding servers. Our scheme has a significantly lower upload cost than the existing schemes, and is naturally scalable to multiplication of an arbitrary number of matrices.
- We describe secure distributed computation schemes for other operations such as matrix addition, matrix transpose, matrix exponentiation, changing the parameters of the secret shares, and solving linear systems, which includes computing the matrix inverse. Thus, it is shown that arbitrary matrix polynomials can be computed securely on distributed servers.

4.1.3 Data representation using finite fields

In order to do meaningful computations on data consisting of real numbers, we need to devise a method to encode real numbers. We discuss below how to map real numbers to finite field elements for performing secure computations. The type of data

representation often determines how fast and efficiently a computation is performed on specific hardware. Special purpose computing servers can be designed for fast finite field computations.

4.1.3.1 Finite field

A finite field $\mathbb{F}_q$ with $q = p^k$ elements can be represented as the quotient polynomial ring $\mathbb{Z}_p[X]/f(x)$, with integer coefficients modulo a prime number p, and the polynomial modulus $f(x)$ chosen to be an irreducible polynomial of degree k in $\mathbb{Z}_p[X]$. The elements of $\mathbb{F}_q$ are polynomials over the field $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ whose degree is strictly less than k.

4.1.3.2 Encoding integers

Given an integer u to encode and an integer base b, a base-b expansion of u is computed, and represented as a polynomial with the same coefficients. The expansion uses a 'balanced' representation of integers modulo b as the coefficients, that is, when b is odd, the coefficients are chosen to lie in the range $\left[\frac{-(b-1)}{2}, \frac{(b-1)}{2}\right]$, and when b is even, the coefficients are chosen to lie in the range $\left[\frac{-b}{2}, \frac{(b-1)}{2}\right]$, except if b is two, when the coefficients are chosen to lie in the set $\{0, 1\}$. For example, if $b = 2$, the integer $26 = 2^4 + 2^3 + 2^1$ is encoded as the polynomial $x^4 + x^3 + x$. If $b = 3$, $26 = 3^3 - 3^0$ is encoded as the polynomial $x^3 - 1$. Decoding is done by computing the polynomial representation at $x = b$. To map an integer to the finite field $\mathbb{F}_q$, where $q = p^k$, the polynomial representation of the integer is viewed as being a polynomial in $\mathbb{F}_q$.

4.1.3.3 Encoding fractional numbers

Real numbers can be approximated by rational numbers to a finite precision. To encode a rational number in a finite field, the number is expanded in a given base b. For example, given $b = 2$, the number $u = 26.75 = 2^4 + 2^3 + 2^1 + 2^{-1} + 2^{-2}$ can be expressed as the polynomial $x^4 + x^3 + x + x^{-1} + x^{-2}$, similarly to the above scheme. If we choose the polynomial modulus to be the irreducible polynomial $f(x) = x^k + 1$,

then the negative degree terms in the polynomial wrap around to the highest degree terms of the polynomial by using the identity $x^k = -1$. That is, the encoding of the number u is now written as

$$26.75 \rightarrow -x^{k-1} - x^{k-2} + x^4 + x^3 + x \tag{4.1}$$

$$= x^{k-1} + x^{k-2} + x^4 + x^3 + x. \tag{4.2}$$

The degree of $f(x)$ is chosen large enough such that there is no "aliasing", that is, the highest degree terms corresponding to the fractional part do not overlap with the lower degree terms corresponding to the integer part.

Rational numbers of fixed precision can also be represented as integers by scaling them with a large enough number, and then encoding them into a finite field similarly to the scheme described for encoding integers.

4.1.3.4 Round-off errors

Consider the rational approximation of real numbers with a precision of $1/m$. The rational numbers $\frac{1}{m}, \frac{2}{m}, \ldots, \frac{j}{m}, j \in \mathbb{Z}$, can be exactly represented as integers by multiplying through by m. However, real numbers in the interval $\left(\frac{j-1}{m}, \frac{j}{m}\right), j \in \mathbb{Z}$, must be rounded-off to the nearest integer multiple of $1/m$ so that they can be represented as integers. This gives rise to round-off errors in digital computations.

4.1.3.5 Computations on real data using finite fields

Integer and rational number arithmetic can be implemented using finite field arithmetic under certain conditions. The addition, subtraction, multiplication and division operations are well defined in finite fields. To make the addition and multiplication in the finite field reflect the addition and multiplication of integers, the result of the computation must not increase the positive or the negative coefficients of the computation result beyond half the coefficient modulus p, and must not increase the number of positive degree terms corresponding to the integer part or the number of negative

degree terms corresponding to the fractional part beyond half the degree of the polynomial modulus $f(x)$. Therefore, the finite field in which computations are implemented should have a much larger coefficient modulus than the base b in which the numbers were represented, and the polynomial modulus must have a degree much larger than the precision of the numbers. As long as the parameters of the finite field encoding are chosen keeping the above constraints in mind, arithmetic on rational number approximations of real numbers can be implemented exactly using finite field arithmetic.

4.2 System Model

We consider Γ source nodes, $N \geq 2$ servers, and one user, for some $\Gamma, N \in \mathbb{N}$ (see Fig. 4.1). Each source node is connected to each server through an orthogonal link. Each pair of servers is connected to each other, and each server is connected to the user through a private link. Each source node $\gamma \in [\Gamma] \triangleq \{1, \ldots, \Gamma\}$ has access to an input matrix $\mathbf{A}^{(\gamma)} \in \mathbb{F}_q^{m_\gamma \times m'_\gamma}$, for $m_\gamma, m'_\gamma \in \mathbb{N}, \gamma \in [\Gamma]$, and the finite field $\mathbb{F}_q$ with q elements. Using N computing servers, the user wants to securely compute the result of a function $\mathbf{C} = G(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(\Gamma)})$, where G is an arbitrary polynomial function, assuming appropriate matrix dimensions. We assume that the entries of $\mathbf{A}^{(\gamma)}, \gamma \in [\Gamma]$, are independent of each other and uniformly distributed over $\mathbb{F}_q$. We also assume that the servers are honest, but curious, which means that each server honestly follows the protocol without spurious insertions, yet may infer information about the inputs passively. Similarly to [70], the system operates in three phases: (1) Sharing, (2) Computation and communication, and (3) Reconstruction. A detailed description of these phases is as follows.

1) Sharing phase: In this phase, the source γ sends secret shares of matrix $\mathbf{A}^{(\gamma)}$, denoted by $[[\mathbf{A}^{(\gamma)}]]_i$, to server i. $[[\mathbf{A}^{(\gamma)}]]_i$ is a function of input matrix $\mathbf{A}^{(\gamma)}$ and a secret key $\mathbf{Sk}^{(\gamma)}$, and its dimensions will depend on the scheme used.

2) Computation and communication: In this phase, the servers process the data they have received from the sources, and may also exchange messages with each

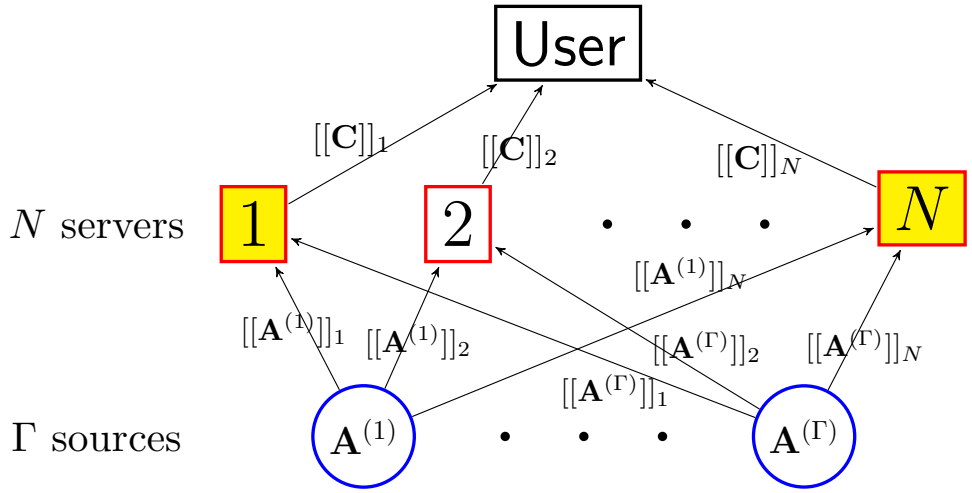


FIGURE 4.1: System model for Γ matrices with N servers, any T of which may collude (shown as shaded servers).

other. We denote the set of all messages that server i sends to server i' in this phase by $\mathcal{M}_{i,i'}$.

3) Reconstruction: In this phase, every server $i \in [N]$ sends a message $[[\mathbf{C}]]_i$ to the user, who decodes the received messages to obtain the desired result of the computation.

The scheme must satisfy the following four constraints.

Correctness The user must be able to decode the final function $\mathbf{C} = G(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(\Gamma)})$ from all the responses $[[\mathbf{C}]]_1, \ldots, [[\mathbf{C}]]_N$. The correctness constraint is imposed by:

$$H\left(\mathbf{C} \middle| [[\mathbf{C}]]_1, \ldots, [[\mathbf{C}]]_N\right) = 0. \tag{4.3}$$

Remark 4.1. *We will also consider the special case of computation with own data, where the data matrices used for computation belong to the user, as in the distributed computation scenario studied in [60, 62, 64, 68]. Accordingly, the correctness constraint becomes:*

$$H\left(\mathbf{C} \middle| [[\mathbf{C}]]_1, \ldots, [[\mathbf{C}]]_N, \mathbf{Sk}^{(1)}, \ldots, \mathbf{Sk}^{(\Gamma)}\right) = 0. \tag{4.4}$$

Security against the servers The goal is to recover $\mathbf{C}$ reliably and securely even if any $T < N$ servers collude to extract some information about the input matrices. Hence, the encoded matrices $[[\mathbf{A}^{(\gamma)}]]_{\mathcal{L}}$, and the set of messages communicated by the servers in $\mathcal{L}^c$ to the servers in $\mathcal{L}$, denoted by $\mathcal{M}_{\mathcal{L}^c,\mathcal{L}}$, $\forall \mathcal{L} \subseteq [N], |\mathcal{L}| \leq T, \gamma \in [\Gamma]$, must not reveal any information about $\mathbf{A}^{(\gamma)}, \forall \gamma \in [\Gamma]$. Accordingly, the security constraint is specified as,

$$I\left(\left\{\mathbf{A}^{(\gamma)}\right\}_{\gamma=1}^{\Gamma}; \left\{[[\mathbf{A}^{(\gamma)}]]_{\mathcal{L}}\right\}_{\gamma=1}^{\Gamma}, \mathcal{M}_{\mathcal{L}^c,\mathcal{L}}\right) = 0, \quad \forall \mathcal{L} \subseteq [N], |\mathcal{L}| \leq T, \forall \gamma \in [\Gamma]. \tag{4.5}$$

Security against the user The user must not gain additional information about the input matrices beyond the result of the function G. This is defined as:

$$I\left(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(\Gamma)}; [[\mathbf{C}]]_1, \ldots, [[\mathbf{C}]]_N \middle| \mathbf{C}\right) = 0.$$

For a given number of servers N and security requirement T, the performance will be measured in terms of the *upload cost* from the source nodes to the servers. The upload cost is defined as follows:

$$\chi_{UL} \triangleq \frac{\sum_{\gamma=1}^{\Gamma} \sum_{i=1}^{N} H([[\mathbf{A}^{(\gamma)}]]_i)}{\sum_{\gamma=1}^{\Gamma} H(\mathbf{A}^{(\gamma)})}, \tag{4.6}$$

and it quantifies the normalized amount of information that must be delivered to the servers. In most cases the amount of computation that must be carried out by each server depends on the amount of information delivered to it, and many cloud computing services charge users based on the amount of information delivered and stored at each server. Hence, minimizing the upload cost will reduce both the latency and the cost of computations. Accordingly, our objective is to securely compute $\mathbf{C}$ by incurring the minimum upload cost. Due to the symmetry across the servers, we will assume (unless stated otherwise) that the secret shares sent from each source to each server is of the same size, and is a $(1/K)^{th}, K \in \mathbb{N}$, fraction of the size of the input matrices.

Definition 4.1. An (N, K, T) SDMC scheme uses N servers, sends $(1/K)^{th}$ fraction

of the input data matrices' size to each server, and is secure against any T colluding servers. Hence, the upload cost of an (N, K, T) SDMC scheme is given by $\chi_{UL} = \frac{N}{K}$.

4.3 SDMM for $\Gamma = 2$

In this section we focus exclusively on the secure distributed multiplication of two matrices, i.e., $\Gamma = 2$. This problem will allow us to present the main ideas behind our coded computation scheme. We first present our main result in the following theorem.

Theorem 4.2. *We can securely multiply two matrices using N servers, T of which may collude, with $N > 2T$, with an upload cost of $\frac{N}{N-2T}$. In other words, an $(N, N-2T, T)$ SDMM scheme for two input matrices is achievable.*

We present the following example to illustrate the essential ingredients of the scheme. For ease of notation, we use the notation $\mathbf{A}^{(1)} = \mathbf{A}$ and $\mathbf{A}^{(2)} = \mathbf{B}$ in this section.

Example: We consider distributed multiplication of matrices $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ and $\mathbf{B} \in \mathbb{F}_q^{n \times p}$ over $N = 7$ servers, any $T = 2$ of which may collude (see Fig. 4.1).

4.3.0.1 Sharing phase

The matrices $\mathbf{A}$ and $\mathbf{B}$ are partitioned into $K = N - 2T = 3$ blocks of dimensions $m \times \frac{n}{3}$ and $\frac{n}{3} \times p$, respectively, as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{A}_3 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \mathbf{B}_3 \end{bmatrix}, \tag{4.7}$$

and the product $\mathbf{C} = \mathbf{AB}$ is given by

$$\mathbf{C} = \mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2 + \mathbf{A}_3\mathbf{B}_3. \tag{4.8}$$

The matrices $\mathbf{R}_i \in \mathbb{F}_q^{m \times \frac{n}{K}}$ and $\mathbf{S}_i \in \mathbb{F}_q^{\frac{n}{K} \times p}, i \in [T]$, are generated, whose entries are independent and identically distributed (i.i.d.) uniform random variables from the

finite field $\mathbb{F}_q$. The matrices $\mathbf{R} = [\mathbf{R}_1, \ldots, \mathbf{R}_T]$ and $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_T]$ are used as the secret keys to encode the input matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. The following two polynomials are constructed to encode the matrices:

$$\mathbf{A}(x) = \mathbf{A}_1 + \mathbf{A}_2 x + \mathbf{A}_3 x^2 + \mathbf{R}_1 x^3 + \mathbf{R}_2 x^4 \tag{4.9}$$

$$\mathbf{B}(x) = \mathbf{B}_1 + \mathbf{B}_2 x^{-1} + \mathbf{B}_3 x^{-2} + \mathbf{S}_1 x^{-5} + \mathbf{S}_2 x^{-6}. \tag{4.10}$$

Note that the encoding polynomials of $\mathbf{A}$ and $\mathbf{B}$ are different from each other, depending on the order of multiplication. We refer to the encoding of matrix $\mathbf{A}$ as "left-encoding", and that of matrix $\mathbf{B}$ as "right-encoding". We refer to each of the N secret shares obtained from left-encoding of matrix $\mathbf{A}$ as an "(N, K, T) left-share", where the $i^{th}, \forall i \in [N]$ secret share is denoted by $[[\mathbf{A}]]_i^L$, and each of those from right-encoding of matrix $\mathbf{B}$ as an "(N, K, T) right-share", where the $i^{th}, \forall i \in [N]$ secret share is denoted by $[[\mathbf{B}]]_i^R$.

Let α_7 be a primitive 7^{th} root of unity in $\mathbb{F}_q$, and $1, \alpha_7, \alpha_7^2, \ldots, \alpha_7^6$ be the 7^{th} roots of unity in $\mathbb{F}_q$. Then the polynomials $\mathbf{A}(x)$ and $\mathbf{B}(x)$ are evaluated at α_7^{i-1} to obtain $[[\mathbf{A}]]_i^L, [[\mathbf{B}]]_i^R$; that is, $[[\mathbf{A}]]_i^L = \mathbf{A}(\alpha_7^{i-1})$ and $[[\mathbf{B}]]_i^R = \mathbf{B}(\alpha_7^{i-1}), i = 1, \ldots, 7$, and the secret shares $[[\mathbf{A}]]_i^L, [[\mathbf{B}]]_i^R$ are sent to server i, $i = 1, \ldots, 7$. The number of symbols sent by the source nodes to the servers is given by $H([[\mathbf{A}]]_i) + H([[\mathbf{B}]]_i) = \frac{mn}{K} + \frac{np}{K}$.

Remark 4.2. *We note here that evaluating the polynomials* $\mathbf{A}(x)$ *and* $\mathbf{B}(x)$ *at the roots of unity is equivalent to computing the discrete Fourier transform of the sequences* $\{\mathbf{A}_1, \ldots, \mathbf{A}_K, \mathbf{R}_1, \ldots, \mathbf{R}_T, \mathbf{0}_1, \ldots, \mathbf{0}_T\}$ *and* $\{\mathbf{B}_1, \ldots, \mathbf{B}_K, \mathbf{0}_1, \ldots, \mathbf{0}_T, \mathbf{S}_1, \ldots, \mathbf{S}_T\}$ *in a finite field, where* $\mathbf{0}_k, k = 1, \ldots, T$, *are zero matrices. This can be carried out efficiently using the FFT algorithm for finite fields introduced in [72].*

4.3.0.2 Computation phase

Server i computes the product

$$[[\mathbf{C}]]_i = [[\mathbf{AB}]]_i = [[\mathbf{A}]]_i^L [[\mathbf{B}]]_i^R, \tag{4.11}$$

which is equivalent to the evaluation of the polynomial

$$\begin{aligned}\mathbf{C}(x) =& \mathbf{A}(x)\mathbf{B}(x) && (4.12)\\ =& \mathbf{A}_1\mathbf{S}_2x^{-6} + (\mathbf{A}_1\mathbf{S}_1 + \mathbf{A}_2\mathbf{S}_2)\, x^{-5} + \cdots \\ &+ (\mathbf{A}_1\mathbf{B}_2 + \mathbf{A}_2\mathbf{B}_3 + \mathbf{R}_2\mathbf{S}_1)\, x^{-1} \\ &+ \left(\sum_{l=1}^{3} \mathbf{A}_l\mathbf{B}_l\right) + \cdots + \mathbf{R}_2\mathbf{B}_1x^4 && (4.13)\end{aligned}$$

at $x = \alpha_7^{i-1}$.

4.3.0.3 Reconstruction phase

The servers send $[[\mathbf{C}]]_i$'s to the user. We know that

$$\sum_{i=1}^{N}(\alpha_N^{i-1})^s = 0, \quad \forall s : N \nmid s. \tag{4.14}$$

Therefore, the user computes the average of the received responses to obtain the final result:

$$\frac{1}{7}\sum_{i=1}^{7}[[\mathbf{C}]]_i = \sum_{l=1}^{3}\mathbf{A}_l\mathbf{B}_l, \tag{4.15}$$

because the non-constant terms from Eq. (4.13) sum to 0 thanks to Eq. (4.14).

For the general case with N servers, T of which can collude, the input matrices are partitioned into $K = N - 2T$ submatrices similarly to Eq. (4.7). Then the product can be written as $\mathbf{C} = \mathbf{AB} = \sum_{l=1}^{K}\mathbf{A}_l\mathbf{B}_l$.

The matrices $\mathbf{A}$ and $\mathbf{B}$ are encoded with the following two polynomials:

$$\mathbf{A}(x) = \sum_{l=1}^{K}\mathbf{A}_l x^{l-1} + \sum_{l=1}^{T}\mathbf{R}_l x^{K+l-1}, \tag{4.16}$$

and

$$\mathbf{B}(x) = \sum_{l=1}^{K} \mathbf{B}_l x^{-l+1} + \sum_{l=1}^{T} \mathbf{S}_l x^{-K-T-l+1}. \tag{4.17}$$

We define the product polynomial as follows.

$$\mathbf{C}(x) = \sum_{l=1}^{K} \mathbf{A}_l \mathbf{B}_l + (\text{non-constant terms}). \tag{4.18}$$

The goal is to recover the constant term in $\mathbf{C}(x)$ from the computations of N servers.

Remark 4.3. *Since the polynomials must be evaluated at the N^{th} roots of unity in the proposed scheme, the finite field must be chosen to guarantee the presence of all the N^{th} order roots of unity. Therefore, we must have $N|(q-1)$. This can be satisfied by appropriately choosing the field size q.*

The polynomials $\mathbf{A}(x)$ and $\mathbf{B}(x)$ are evaluated at the N^{th} roots of unity, denoted by $1, \alpha_N, \alpha_N^2, \ldots, \alpha_N^{N-1} \in \mathbb{F}_q$, where α_N is a primitive N^{th} root of unity in $\mathbb{F}_q$. Thus, the values sent to server i are $[[\mathbf{A}]]_i^L = \mathbf{A}(\alpha_N^{i-1})$ and $[[\mathbf{B}]]_i^R = \mathbf{B}(\alpha_N^{i-1})$. Server $i \in [N]$ computes the share $[[\mathbf{C}]]_i = [[\mathbf{A}]]_i^L [[\mathbf{B}]]_i^R = \mathbf{C}(\alpha_N^{i-1})$, and sends it to the user.

Thanks to Eq. (4.14), the user obtains the desired result by averaging the received $[[\mathbf{C}]]_i$'s.

$$\frac{1}{N} \sum_{i=1}^{N} [[\mathbf{C}]]_i = \sum_{l=1}^{K} \mathbf{A}_l \mathbf{B}_l. \tag{4.19}$$

4.3.1 Special case: Computation with own data

For the special case in which the data matrices belong to the user, the upload cost can be further reduced since the user also has access to the random secret keys used for generating the secret shares. Most previous literature on SDMM considers this special case. The user partitions the input matrices into K blocks as in Eq. (4.7), where

$K = N - T$. The user then encodes the matrices using the following polynomials.

$$\mathbf{A}(x) = \sum_{l=1}^{K} \mathbf{A}_l x^{l-1} + \sum_{l=1}^{T} \mathbf{R}_l x^{K+l-1}, \tag{4.20}$$

and

$$\mathbf{B}(x) = \sum_{l=1}^{K} \mathbf{B}_l x^{-l+1} + \sum_{l=1}^{T} \mathbf{S}_l x^{-K-l+1}, \tag{4.21}$$

where $\mathbf{A}(x)$ is the same as in Eq. (4.16) while we have a slight change from Eq. (4.17) in the way the secret key is embedded into the $\mathbf{B}(x)$ polynomial. The user evaluates the polynomials on the N^{th} roots of unity to generate the secret shares, and sends these shares to the servers. The servers return their computed results back to the user, where the constant term in the product polynomial $\mathbf{C}(x)$ is now given by $\sum_{i=1}^{K} \mathbf{A}_i\mathbf{B}_i + \sum_{i=1}^{T} \mathbf{R}_i\mathbf{S}_i$. The user then averages the received results to obtain

$$\frac{1}{N}\sum_{i=1}^{N}[[\mathbf{C}]]_i = \sum_{l=1}^{K} \mathbf{A}_l\mathbf{B}_l + \sum_{l=1}^{T} \mathbf{R}_l\mathbf{S}_l. \tag{4.22}$$

Since the user has access to the secret keys it has used to encrypt the partitions, it can subtract their product, $\sum_{i=1}^{T} \mathbf{R}_i\mathbf{S}_i$, to obtain the desired result. We note here that the product of the random matrices $\mathbf{R}$ and $\mathbf{S}$ needs to be pre-computed by the user in order to obtain the desired matrix product from Eq. (4.22). If $T << N-T$, this would require much less computational resources compared to multiplying the matrices $\mathbf{A}$ and $\mathbf{B}$, or alternatively these multiplications can be done in advance in an offline manner, and stored at the user, and hence, this computation does not affect the computation latency.

Theorem 4.3. *A $(N, N-T, T)$ SDMM scheme for two input matrices is achievable for the special case, or in other words, we can securely multiply two matrices using N servers, T of which may collude, with $N > T$, with an optimal upload cost of $\frac{N}{N-T}$.*

Proof. It follows from the definition of an SDMM scheme that the upload cost of the $(N, N-T, T)$ SDMM scheme proposed above is $\chi_{UL} = \frac{N}{N-T}$. Moreover, it is proved

in [95] that the optimal upload cost of an SDMM scheme is lower bounded by $\frac{N}{N-T}$. This proves the optimality of the proposed scheme in terms of the upload cost. □

4.3.2 Proof of security against the servers

We next prove that the proposed scheme is secure, i.e., the security constraint (4.5) is satisfied. We point out that there is no exchange of messages between the servers, that is, $\mathcal{M}_{i,i'} = \phi$ for all $i, i' \in [N], i \neq i'$. For any $\mathcal{L}$ with $|\mathcal{L}| = T$, we have

$$
\begin{aligned}
I(\mathbf{A},\mathbf{B};[[\mathbf{A}]]_{\mathcal{L}},[[\mathbf{B}]]_{\mathcal{L}}) &= H([[\mathbf{A}]]_{\mathcal{L}},[[\mathbf{B}]]_{\mathcal{L}}) && (4.23)\\
&\quad - H([[\mathbf{A}]]_{\mathcal{L}},[[\mathbf{B}]]_{\mathcal{L}}|\mathbf{A},\mathbf{B}) && (4.24)\\
&\overset{a}{=} H([[\mathbf{A}]]_{\mathcal{L}},[[\mathbf{B}]]_{\mathcal{L}}) - H(\mathbf{R},\mathbf{S}) && (4.25)\\
&\overset{b}{=} H([[\mathbf{A}]]_{\mathcal{L}}) + H([[\mathbf{B}]]_{\mathcal{L}}) - H(\mathbf{R}) - H(\mathbf{S}) && (4.26)\\
&\leq \sum_{i\in\mathcal{L}} H([[\mathbf{A}]]_i) + \sum_{i\in\mathcal{L}} H([[\mathbf{B}]]_i)\\
&\qquad - \frac{mnT}{K}\log|\mathbb{F}_q| - \frac{pnT}{K}\log|\mathbb{F}_q| && (4.27)\\
&= 0, && (4.28)
\end{aligned}
$$

where (a) follows from (4.16) and (4.17); (b) follows from the fact that $[[\mathbf{A}]]_{\mathcal{L}}$, $[[\mathbf{B}]]_{\mathcal{L}}$, $\mathbf{R}$ and $\mathbf{S}$ are independent of each other; (4.28) follows because the elements of the secret shares $[[\mathbf{A}]]_i \in \mathbb{F}_q^{m\times\frac{n}{K}}$ and $[[\mathbf{B}]]_i \in \mathbb{F}_q^{\frac{n}{K}\times p}, \forall i \in \mathcal{L}$ are independent and uniformly distributed in $\mathbb{F}_q$. Hence, $\sum_{i\in\mathcal{L}} H([[\mathbf{A}]]_i) = \frac{mnT}{K}\log|\mathbb{F}_q|$ and $\sum_{i\in\mathcal{L}} H([[\mathbf{B}]]_i) = \frac{pnT}{K}\log|\mathbb{F}_q|$, where $|\mathbb{F}_q|$ denotes the cardinality of the field $\mathbb{F}_q$.

4.3.3 Security against the user

When the input matrices are generated by distributed source nodes, the user must not gain additional information beyond the result of the computation. Most existing schemes [60, 62, 64, 68] which rely on polynomial interpolation, do not satisfy this condition, since they are designed for the case when the data is generated by the user. If those schemes are employed, once $\mathbf{C}(x) = \mathbf{A}(x)\mathbf{B}(x)$ is interpolated by the user after receiving the evaluations of $\mathbf{C}(x)$ at a number of points equal to the number

of terms in $\mathbf{C}(x)$, it can be factorized to obtain information about $\mathbf{A}(x)$ and $\mathbf{B}(x)$, thus leaking additional information to the user. Our scheme is robust to such information leakages.

We point out that besides the constant term, the user can also compute the sum of the coefficients of x^i and $x^{-i}, i \in [N-1]$, in Eq. (4.13). We observe in Eq. (4.13) that the non-constant terms are uniformly distributed in $\mathbb{F}_q^{m\times p}$ and are independent, so that no information can be gained about $\mathbf{A}$ or $\mathbf{B}$ from computing those terms. Since only N evaluations of $\mathbf{C}(x)$, which is of degree $2N-1$, are received by the user, it cannot be interpolated, and indeed, the user cannot gain any information about the input matrices beyond their product [63].

4.3.4 Other performance metrics

4.3.4.1 Encoding complexity

To compute N evaluations of the matrix polynomial $\mathbf{A}(x)$ on the roots of unity, the source nodes perform $\frac{mn}{K}$ N-point FFTs, which involve $O(\frac{mnN\log N}{K})$ finite field operations in $\mathbb{F}_q$. For $T=0$, the complexity is $O(mn\log N)$, that is, it is a logarithmic rate of growth with respect to N. The analysis is similar for computing the evaluations of matrix polynomial $\mathbf{B}(x)$.

4.3.4.2 Download cost

Download cost is the normalized number of bits that need to be downloaded by the user from the servers to reconstruct the computation result. It is defined as

$$\chi_{DL} = \frac{\sum_{i=1}^{N} H([[\mathbf{C}]]_i)}{H(\mathbf{C})}. \tag{4.29}$$

As shown in [65], the download cost depends on the dimensions of the data matrices. The following lemma from [65] determines the entropy of the product of two matrices depending on their dimensions.

Lemma 4.1. *Let* $\mathbf{A}, \mathbf{B}$ *be random matrices independently and uniformly distributed over* $\mathbb{F}_q^{m\times n}$ *and* $\mathbb{F}_q^{n\times p}$*, respectively. As* $q \to \infty$*, we have*

$$H(\mathbf{AB}) = \begin{cases} mp & n \geq \min(m,p) \\ mn + np - n^2 & n < \min(m,p) \end{cases}, \tag{4.30}$$

in $q-$*ary units.*

Therefore, we can compute the download cost of our scheme for three different cases.

- If $\min(m,p) \leq n \leq K\min(m,p)$, the download cost is

$$\chi_{DL} = \frac{N(\frac{mn}{K} + \frac{np}{K} - \frac{n^2}{K^2})}{mp} \tag{4.31}$$

$$= \frac{Nn(m+p-\frac{n}{K})}{Kmp}. \tag{4.32}$$

  Assuming $m = p$, we have $\chi_{DL} \approx \frac{2nN}{mK} < \frac{2N}{K}$.

- If $n \geq K\min(m,p)$, the download cost is

$$\chi_{DL} = \frac{Nmp}{mp} \tag{4.33}$$

$$= N. \tag{4.34}$$

- If $n < \min(m,p)$, the download cost is

$$\chi_{DL} = \frac{N(\frac{mn}{K} + \frac{np}{K} - \frac{n^2}{K^2})}{mn + np - n^2} \tag{4.35}$$

$$= \frac{N(m+p-\frac{n}{K})}{K(m+p-n)}. \tag{4.36}$$

  For $m, p \to \infty$, $\chi_{DL} \to \frac{N}{K} = \frac{N}{N-T}$. This is the optimal download cost.

4.3.4.3 Decoding complexity

Since the decoding requires computing the sum of the received results, the decoding complexity of the proposed scheme is negligible. This is an important advantage of the

proposed scheme compared to existing polynomial coding schemes in the literature, which require polynomial interpolation.

4.3.5 Comparison with other schemes

There exists a trade-off between the upload cost and the download cost depending on the kind of partitioning employed for matrix multiplication. In [95, 96], the trade-off within the class of schemes that employ row-by-column partitioning is considered. The other class of schemes that employ column-by-row partitioning, like MatDot and our scheme, provide different points on the upload cost-download cost trade-off. The GASP [64], secure PolyDot [68], and USCSA [95] schemes, which employ row-by-column partitioning generally result in a higher upload cost and lower download cost than those employing column-by-row partitioning. Consider K row-wise partitions of $\mathbf{A}$, and K column-wise partitions of $\mathbf{B}$. For $T = 0$, the matrices are encoded by evaluating their corresponding polynomials at $N = K^2$ distinct points in $\mathbb{F}_q$. SDMM schemes employing the row-by-column partitioning require $N = O(K^2)$ servers. Hence, the upload cost is $\chi_{UL} = \frac{N}{K} = O(\frac{N}{\sqrt{N}}) = O(\sqrt{N})$. Since the complexity of evaluating a polynomial at N points is $O(N \log N^2 \log\log N)$ [97], the total encoding complexity is $O(\frac{mn}{\sqrt{N}} N \log^2 N \log\log N) \approx O(mn\sqrt{N} \log^2 N \log\log N)$. This is significantly larger than the encoding complexity of our scheme, which grows only logarithmically with N.

The secure MatDot scheme of [68], which employs column-by-row partitioning, has an upload cost of $\chi_{UL} = \frac{2N}{N-2T+1}$, which is also larger than that of our scheme by a factor of two (see Fig. 4.2).

Compared to [70], which also considers the general case with the source nodes separate from the user, our scheme provides significant improvements in terms of the upload cost. The upload cost achieved by the scheme in [70] is given by $\chi_{UL} = \frac{\min(2K^2+2T-3, K^2+KT+T-2)}{K}$, which is significantly higher than that provided by our scheme (see Fig. 4.2). We also highlight that, as opposed to the scheme in [70], our scheme does not require inter-server communication in the computation and communication phases for multiplying two matrices, which significantly reduces both the latency

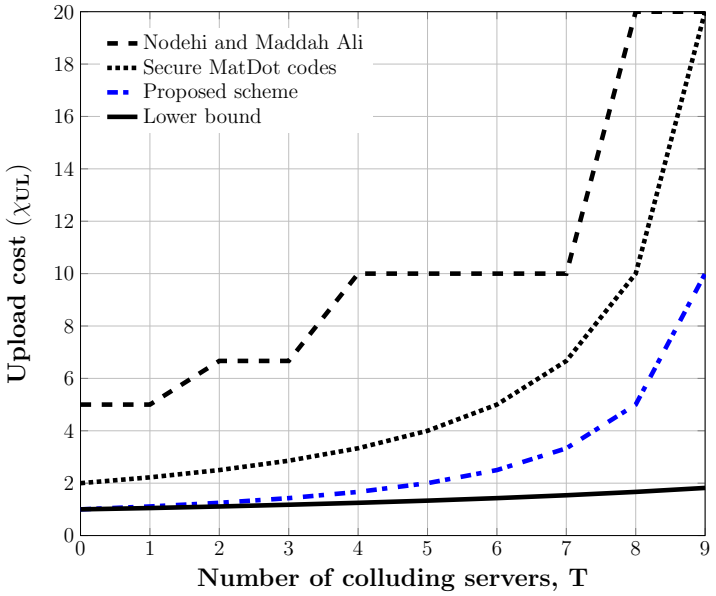


FIGURE 4.2: Comparison of the upload cost of SDMM of two matrices for different number of colluding servers out of a total of $N = 20$ servers. Only the values of $T \leq 9$ are considered since the proposed scheme is defined only for $T < N/2$. The maximum value of K is calculated for each T.

and the complexity.

4.4 Straggler Mitigation

The FFT-based scheme in the previous sections does not provide robustness against straggling servers. We need all the evaluations from all the servers to recover the desired result. In this section we present an extension of our FFT-based scheme, which provides a certain level of robustness to straggling servers by incorporating both column-wise and row-wise partitioning of the matrices.

When providing straggler mitigation, typically the goal is to minimize the *recovery threshold*, which refers to the minimum number of responsive servers. Hence, instead of specifying the number of servers, and minimizing the upload cost, in this section, we consider arbitrary partitioning of the matrices, and identify the corresponding upload cost and the recovery threshold.

In particular, we employ row-wise partitioning of $\mathbf{A}$ (and column-wise for $\mathbf{B}$) to

introduce straggler-robustness. That is, $\mathbf{A}$ is partitioned into a $K_2 \times K_1$ array of equal-sized blocks, and $\mathbf{B}$ is partitioned into a $K_1 \times K_3$ array of equal-sized blocks. The idea is that the secret shares of each row of $\mathbf{A}$ (and each column of $\mathbf{B}$) are generated, while polynomially-coded shares of each column of $\mathbf{A}$ (and each row of $\mathbf{B}$) are generated for straggler robustness. The proposed scheme requires $N \geq K_2K_3(K_1 + 2T)$ responsive servers in general, and $N \geq K_2K_3(K_1 + T)$ responsive servers for computing with own data.

The matrices $\mathbf{A}$ and $\mathbf{B}$ are partitioned into K_1K_2 and K_2K_3 blocks, denoted by $\mathbf{A}_{i,j}, i \in [K_2], j \in [K_1]$ and $\mathbf{B}_{j,k}, j \in [K_1], k \in [K_3]$, respectively. Matrices $\mathbf{R}_{i,j}, i \in [K_2], j \in [T]$, and $\mathbf{S}_{j,k}, j \in [T], k \in [K_3]$, are generated, whose entries are i.i.d. uniform random variables from $\mathbb{F}_q$. The matrices $\mathbf{R}$ and $\mathbf{S}$, defined as

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{1,1} & \cdots & \mathbf{R}_{1,T} \\ \vdots & \ddots & \vdots \\ \mathbf{R}_{K_2,1} & \cdots & \mathbf{R}_{K_2,T} \end{bmatrix}, \tag{4.37}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{1,1} & \cdots & \mathbf{S}_{1,K_3} \\ \vdots & \ddots & \vdots \\ \mathbf{S}_{T,1} & \cdots & \mathbf{S}_{T,K_3} \end{bmatrix}, \tag{4.38}$$

are used as secret keys to encode the matrices $\mathbf{A}$ and $\mathbf{B}$, respectively, by first appending them in the following manner:

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{R} \end{bmatrix}, \tag{4.39}$$

$$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{B} \\ \mathbf{S} \end{bmatrix}, \tag{4.40}$$

and then constructing the following multivariate polynomials:

$$\mathbf{A}(x_1, x_2) = \sum_{i=1}^{K_2} \sum_{j=1}^{K_1} \mathbf{A}_{i,j} x_2^{i-1} x_1^{j-1} + \sum_{i=1}^{K_2} \sum_{j=1}^{T} \mathbf{R}_{i,j} x_2^{i-1} x_1^{K_1+j-1}$$

$$\mathbf{B}(x_1, x_2) = \sum_{j=1}^{K_1} \sum_{k=1}^{K_3} \mathbf{B}_{j,k} x_1^{-(j-1)} x_2^{(k-1)K_2} + \sum_{j=1}^{T} \sum_{k=1}^{K_3} \mathbf{S}_{j,k} x_1^{-(K_1+T+j-1)} x_2^{(k-1)K_2}.$$

The product polynomial $\mathbf{C}(x_1, x_2)$ is given by

$$\mathbf{C}(x_1, x_2) = \sum_{i=1}^{K_2} \sum_{k=1}^{K_3} \left(\sum_{j=1}^{K_1} \mathbf{A}_{i,j} \mathbf{B}_{j,k} \right) x_2^{K_2(k-1)+i-1} + (\text{terms with non-zero powers of } x_1). \tag{4.41}$$

4.4.1 Sharing phase

Let there be a total of $N \geq N_1 N_2$ servers, where $N_1 = K_1 + 2T$ and $N_2 \geq K_2 K_3$. The user evaluates polynomials $\mathbf{A}(x_1, x_2)$ and $\mathbf{B}(x_1, x_2)$ at points $(x_1, x_2) = (\alpha_{N_1}^r, \beta_s), r = 0, \ldots, N_1 - 1, s = 0, \ldots, N_2 - 1$, where α_{N_1} is a primitive N_1^{th} root of unity in $\mathbb{F}_q$, and $\beta_s, \forall s \in [N_2]$ are distinct points in $\mathbb{F}_q$.

The user sends $\mathbf{A}(\alpha_{N_1}^r, \beta_s)$ and $\mathbf{B}(\alpha_{N_1}^r, \beta_s)$ to server i, where $i = (s-1)N_1 + r - 1, \forall r \in [N_1], \forall s \in [N_2]$.

4.4.2 Computation phase

Server i, where $i = (s-1)N_1 + r - 1$, computes $\mathbf{C}(\alpha_{N_1}^r, \beta_s) = \mathbf{A}(\alpha_{N_1}^r, \beta_s)\mathbf{B}(\alpha_{N_1}^r, \beta_s)$, and sends this share to the user as soon as it is computed.

4.4.3 Reconstruction phase

The user collects and computes the averages of all the evaluations of $\mathbf{C}(x_1, x_2)$ on the points having the same x_2 coordinate, which removes the terms with non-zero exponents of x_1. Therefore, for every $x_2 \in [N_2]$, the user obtains

$$\mathbf{f}(x_2) = \frac{1}{N_1} \sum_{i=0}^{N_1-1} \mathbf{C}(\alpha_{N_1}^i, x_2) \tag{4.42}$$

$$= \sum_{i=1}^{K_2} \sum_{k=1}^{K_3} \left(\sum_{j=1}^{K_1} \mathbf{A}_{i,j} \mathbf{B}_{j,k} \right) x_2^{K_2(k-1)+i-1}. \tag{4.43}$$

The user then interpolates the polynomial $\mathbf{f}(x_2)$ from any K_2K_3 evaluations to obtain $\sum_{j=1}^{K_1} \mathbf{A}_{i,j}\mathbf{B}_{j,k}$ for all $(i,k) \in [K_2] \times [K_3]$, therefore obtaining the final result $\mathbf{C} = \mathbf{AB}$.

Remark 4.4. *Note that, to reconstruct the desired result, the user requires all N_1 evaluations on the points (x_1, x_2) for $x_1 = 1, \alpha_{N_1}, \ldots, \alpha_{N_1}^{N_1-1}$ and constant x_2. In other words, for any K_2K_3 values of s, the results from all the groups of servers $[(s-1)N_1 : sN_1 - 1]$ are necessary and sufficient for reconstructing the desired result. Therefore, this scheme provides a slightly weaker 'group-wise' straggler-robustness.*

4.4.4 Special case: computation with own data

For the special case when the data matrices belong to the user, the encoding polynomials are constructed by the user as follows.

$$\mathbf{A}(x_1, x_2) = \sum_{i=1}^{K_2} \sum_{j=1}^{K_1} \mathbf{A}_{i,j} x_2^{i-1} x_1^{j-1} + \sum_{i=1}^{K_2} \sum_{j=1}^{T} \mathbf{R}_{i,j} x_2^{i-1} x_1^{K_1+j-1} \tag{4.44}$$

$$\mathbf{B}(x_1, x_2) = \sum_{j=1}^{K_1} \sum_{k=1}^{K_3} \mathbf{B}_{j,k} x_1^{-(j-1)} x_2^{K_2(k-1)} + \sum_{j=1}^{T} \sum_{k=1}^{K_3} \mathbf{S}_{j,k} x_1^{-(K_1+j-1)} x_2^{K_2(k-1)}. \tag{4.45}$$

By this construction, the matrices $\mathbf{R}$ and $\mathbf{S}$ also appear in the constant coefficient of the polynomials $\mathbf{f}(x_2)$, and the user must pre-compute the products of the private random matrices in order to obtain the desired result.

4.4.5 Performance Analysis:

- **Upload Cost:** We have $\chi_{UL} = \frac{N(K_3mn + K_2np)}{K_1K_2K_3(mn+np)}$.
- **Encoding complexity:** The complexity of encoding matrix $\mathbf{A}$ for N servers is $O\left(\frac{mn}{K_1K_2}\left(N_1 \log N_1\right)\left(N_2 \log^2 N_2 \log\log N_2\right)\right)$. The encoding complexity for matrix $\mathbf{B}$ can be computed similarly.

- **Decoding complexity:** The decoding requires addition, which has negligible complexity compared to multiplication in a finite field, followed by interpolation of a polynomial with K_2K_3 terms, which has complexity $O(n\log^2 n\log\log n)$ for $n = K_2K_3$ [97]. Note that the decoding complexity of secure PolyDot scheme in [68] is given by the complexity of interpolating a polynomial with $K_2K_3(K_1+T)+K_2K_1+K_2T-1$ terms, while our scheme requires interpolating a polynomial with K_2K_3 terms, which is an order of magnitude smaller.

- **Field size:** The field must have the N_1^{th} roots of unity, and must be big enough to have K_2K_3 distinct elements. Therefore, $N_1|q-1$ and $q > K_2K_3$ must be satisfied.

4.5 Secure Distributed Multiplication of Multiple Matrices

In this section, we consider the multiplication of multiple matrices. Such computations, known as matrix chain multiplications, occur in many applications in signal processing, graph theory, and network analysis. We extend our scheme proposed in Section 4.3 to implement multiplication of multiple matrices, that is, given Γ source nodes generating the matrices $\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(\Gamma)}$, the user wants to obtain the product $\mathbf{C} = G(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(\Gamma)}) = \mathbf{A}^{(1)}\cdots\mathbf{A}^{(\Gamma)}$ securely from distributed computation over N available servers. A naive method would be for the servers to securely compute the multiplication of two matrices at a time using the proposed scheme in Section 4.3, and send the results of the computations to the user so that he reconstructs the intermediate computation results, that is, the product of a subset of the matrices, before re-encoding the intermediate result and sending its secret shares to the servers to multiply with the next matrix. However, such a naive method incurs an unnecessary amount of communication cost between the user and the servers, and also requires the user to re-encode the intermediate matrices multiple times, thus increasing the latency of the computation. The naive method also leaks information about the intermediate computations to the user. Next, we propose a more efficient alternative, that also satisfies the user privacy constraint.

The following scheme proceeds iteratively in multiple rounds by obtaining the shares of $\mathbf{C}^{(\gamma)} \triangleq \mathbf{A}^{(1)} \cdots \mathbf{A}^{(\gamma)}$, denoted by $[[\mathbf{C}^{(\gamma)}]]_i \triangleq [[\mathbf{A}^{(1)} \cdots \mathbf{A}^{(\gamma)}]]_i, \forall i \in [N]$, in the $(\gamma-1)^{th}$ round, for $\gamma \in [\Gamma]$.

4.5.1 Sharing phase

For $K = N-2T$, the $(N, N-2T, T)$ left-shares $[[\mathbf{A}^{(1)}]]_i^L$ and right-shares $[[\mathbf{A}^{(\gamma)}]]_i^R$, $\forall \gamma \in [2:\Gamma]$, are sent to server i.

4.5.2 Computation phase

In the computation phase of the γ^{th} round, where $\gamma \in [\Gamma-1]$, server $i \in [N]$ computes the shares $[[\mathbf{H}^{(\gamma+1)}]]_i = [[\mathbf{C}^{(\gamma)}]]_i^L [[\mathbf{A}^{(\gamma+1)}]]_i^R$, $\forall \gamma \in [\Gamma-1]$. The secret shares $[[\mathbf{H}^{(\gamma+1)}]]_i$ are evaluations on the N^{th} roots of unity of a polynomial $\mathbf{H}^{(\gamma+1)}(x)$, which is similar to the product polynomial obtained in Eq. (4.13), and whose constant term is the matrix $\mathbf{C}^{(\gamma+1)} = \mathbf{C}^{(\gamma)}\mathbf{A}^{(\gamma+1)}$, while the remaining terms are uniformly distributed random matrices.

4.5.3 Communication phase

In the communication phase, the servers exchange shares of their results from the computation phase in a secure way, similarly to [70], to convert their secret shares $[[\mathbf{H}^{(\gamma+1)}]]$ to $(N, N-2T, T)$ left-shares of matrix $\mathbf{C}^{(\gamma+1)}$. To do this,

- Server i generates $(N, N-2T, T)$ left-shares of $[[\mathbf{H}^{(\gamma+1)}]]_i$, evaluated on the N^{th} roots of unity $\alpha_N^{j-1}, \forall j \in [N]$, with α_N being a primitive N^{th} root of unity in $\mathbb{F}_q$, and enumerated as $[[\mathbf{H}^{(\gamma+1)}]]_{i,j}^L = \left[\left[[[\mathbf{H}^{(\gamma+1)}]]_i\right]\right]_j^L, \forall j \in [N]$.
- Server i sends the left-share $[[\mathbf{H}^{(\gamma+1)}]]_{i,j}^L$ to server j. The privacy requirement against the servers is satisfied, since any T colluding servers cannot gain any information about server i's share $[[\mathbf{H}^{(\gamma+1)}]]_i$ from the left-shares received in the communication phase.

- Server j averages the received left-shares $[[\mathbf{H}^{(\gamma+1)}]]^L_{i,j}, \forall i \in [N]$, to obtain the $(N, N-2T, T)$ left-shares $[[\mathbf{C}^{(\gamma+1)}]]^L_j$.

To see the correctness of the above procedure, note that, for given N, K and T values, the secret sharing scheme is linear for both left and right shares; that is, $[[\mathbf{A}]] + [[\mathbf{B}]] = [[\mathbf{A}+\mathbf{B}]]$. Therefore, following from Eq. (4.19), we have

$$\frac{1}{N}\sum_{i=1}^{N}[[\mathbf{H}^{(\gamma+1)}]]^L_{i,j} = \frac{1}{N}\sum_{i=1}^{N}\left[\left[[[\mathbf{H}^{(\gamma+1)}]]_i\right]\right]^L_j \tag{4.46}$$

$$= \left[\left[\mathbf{C}^{(\gamma+1)}\right]\right]^L_j. \tag{4.47}$$

The scheme proceeds in a recursive manner, looping back to the computation phase for the $(\gamma+1)^{th}$ round.

4.5.4 Reconstruction phase

At the end of the $(\Gamma-1)^{th}$ round, the servers have access to the secret shares of the matrix $\mathbf{C}^{(\Gamma)} = \mathbf{A}^{(1)}\cdots\mathbf{A}^{(\Gamma)}$. The servers send the shares $[[\mathbf{C}^{(\Gamma)}]]_i, \forall i \in [N]$, to the user, which then computes the average of these shares to obtain

$$\frac{1}{N}\sum_{i=1}^{N}[[\mathbf{C}^{(\Gamma)}]]_i = \mathbf{A}^{(1)}\cdots\mathbf{A}^{(\Gamma)}. \tag{4.48}$$

4.5.5 Performance analysis

Upload cost: Each server receives one $(N, N-2T, T)$ share of all $\mathbf{A}^{(\gamma)}, \gamma \in [\Gamma]$. Thus we have

$$\chi_{UL} = \frac{\sum_{i=1}^{N}\sum_{\gamma=1}^{\Gamma}\frac{H([[\mathbf{A}^{(\gamma)}]]_i)}{K}}{\sum_{\gamma=1}^{\Gamma}H(\mathbf{A}^{(\gamma)})} \tag{4.49}$$

$$= \frac{N}{N-2T}. \tag{4.50}$$

Encoding complexity: Since the encoding involves the computation of FFT, the complexity is $O(m_\gamma m'_\gamma \frac{N}{K} \log N)$ for encoding the matrix $\mathbf{A}^{(\gamma)}, \forall \gamma \in [\Gamma]$.

Decoding complexity: The partial decoding in the communication phase of each intermediate round, as well as the final decoding of the result after the completion of the $(\Gamma - 1)^{th}$ round requires only the addition of the received results from all the servers. Therefore, the decoding complexity is negligible.

Privacy against the user: The scheme constructed in this section preserves privacy against the user, since the intermediate computation results are not communicated to the user in any form. In contrast, this privacy constraint is violated by the naive scheme described at the start of this section, because the user obtains the result of each intermediate computation.

4.6 Secure Matrix Algebra

In this section, we describe algorithms for performing matrix operations besides matrix multiplication that are useful in matrix algebra. Some operations, like matrix inversion, can be reduced to matrix multiplication, and can be implemented with the SDMM schemes described in the preceding sections.

4.6.1 Matrix addition and multiplication by a scalar

Addition and scalar multiplication follow easily from the linear nature of the secret sharing scheme. We have $[[\mathbf{A} + \mathbf{B}]] = [[\mathbf{A}]] + [[\mathbf{B}]]$, and $[[c\mathbf{A}]] = c[[\mathbf{A}]]$, where $c \in \mathbb{F}_q$. For addition, the shares must both be either left-shares or right-shares.

4.6.2 Changing from (N, K_1, T_1) left-shares to (N, K_2, T_2) right-shares

Suppose the servers store (N, K_1, T_1) left-shares of matrix $\mathbf{A}$. The goal is to let the servers obtain (N, K_2, T_2) right-shares of matrix $\mathbf{A}$. A special case of this algorithm with $K_1 = 1$, $K_2 = N - 2T$ and $T_1 = T_2 = T$ was used earlier in the communication phase of Section 4.5. The procedure takes the following steps (see Fig. 4.3):

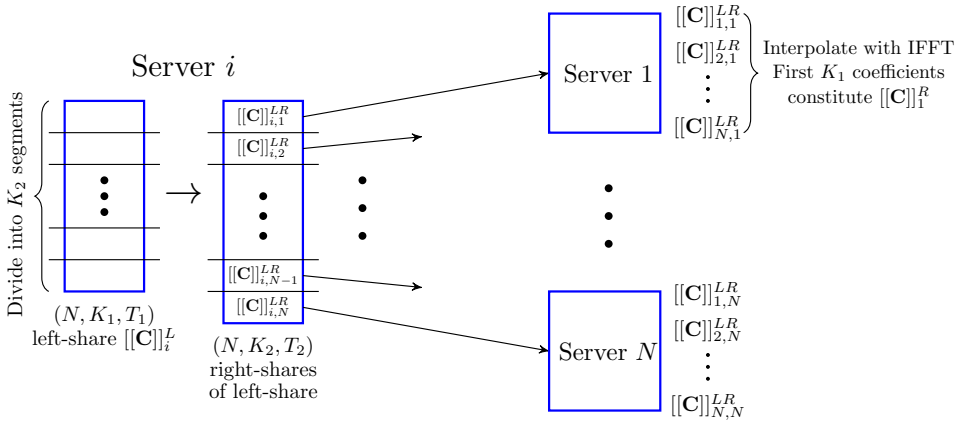


FIGURE 4.3: Conversion from left-share to right-share (right-share to left-share conversion can be done similarly).

1. Server i generates (N, K_2, T_2) right-shares of $[[\mathbf{A}]]_i^L$ evaluated on the N^{th} roots of unity $\alpha_N^{j-1}, \forall j \in [N]$, and enumerated as $[[\mathbf{A}]]_{i,j}^{LR} = \left[\left[[[\mathbf{A}]]_i^L\right]\right]_j^R, \forall j \in [N]$.

2. Server $i \in [N]$ sends the right-share $[[\mathbf{A}]]_{i,j}^{LR}$ to server j.

3. Server j interpolates the received shares $[[\mathbf{A}]]_{i,j}^{LR}, i = 1, \ldots, N$, using inverse FFT (IFFT) to obtain a polynomial whose first K_1 coefficients are K_1 column-wise partitions of the share $[[\mathbf{A}]]_j^R$. Stacking them column-wise gives the share $[[\mathbf{A}]]_j^R$.

A procedure to convert right-shares to left-shares can be obtained similarly to the above procedure. For example, to convert right-shares to left-shares, step 1 of the above procedure generates (N, K_2, T_2) left-shares of $[[\mathbf{A}]]_i^R$, enumerated as $[[\mathbf{A}]]_{i,j}^{RL} = \left[\left[[[\mathbf{A}]]_i^R\right]\right]_j^L, \forall j \in [N]$.

To see the correctness of the above procedure, note that IFFT of a sequence of (N, K_1, T_1) left-shares $\left\{[[\mathbf{A}]]_1^L, \ldots, [[\mathbf{A}]]_N^L\right\}$ gives the sequence $\{\mathbf{A}_1, \ldots, \mathbf{A}_{K_1}, \mathbf{R}_1, \ldots, \mathbf{R}_{T_1}, \mathbf{0}_1, \ldots, \mathbf{0}_{T_1}\}$. Similarly, IFFT of a sequence of (N, K_1, T_1) right-shares $\left\{[[\mathbf{A}]]_1^R, \ldots, [[\mathbf{A}]]_N^R\right\}$ gives the sequence $\{\mathbf{A}_1, \ldots, \mathbf{A}_{K_1}, \mathbf{0}_1, \ldots, \mathbf{0}_{T_1}, \mathbf{R}_1, \ldots, \mathbf{R}_{T_1}\}$, where the subscript $l \in [T_1]$ in $\mathbf{0}_l$ is added only to count the number of zero matrices. Since each coefficient of the IFFT sequence is a linear combination of the elements of the input sequence; for all $l \in [K_1]$, the l^{th} coefficient of the IFFT, denoted by $\text{IFFT}_l(.)$, of the sequence of

shares received by server j is given by

$$\begin{aligned}
&\mathrm{IFFT}_l\Big([[\mathbf{A}]]_{1,j}^{LR},\ldots,[[\mathbf{A}]]_{N,j}^{LR}\Big) && (4.51)\\
&\quad=\mathrm{IFFT}_l\Big(\big[\big[[[\mathbf{A}]]_1^L\big]\big]_j^R,\ldots,\big[\big[[[\mathbf{A}]]_N^L\big]\big]_j^R\Big) && (4.52)\\
&\quad=\Big[\Big[\mathrm{IFFT}_l\left([[\mathbf{A}]]_1^L,\ldots,[[\mathbf{A}]]_N^L\right)\Big]\Big]_j^R && (4.53)\\
&\quad=[[\mathbf{A}_l]]_j^R, && (4.54)
\end{aligned}$$

where $\mathbf{A}_l, l\in[K_1]$ is the l^{th} column-wise partition used for obtaining the original (N,K_1,T_1) left-shares of matrix $\mathbf{A}$. Thus, server j obtains the secret shares $[[\mathbf{A}_l]]_j^R, l=1,\ldots,K_1$.

Remark 4.5. *For $K_1\geq 2$, left-shares cannot be directly converted to left-shares using the above procedure. Similarly, right-shares cannot be directly converted to right-shares. However, $(N,1,T_1)$ left-shares can be directly converted to both (N,K_2,T_2) left-shares and right-shares using the above procedure. For converting $(N,1,T_1)$ left-shares to (N,K_2,T_2) left-shares, step 1 of the above procedure generates (N,K_2,T_2) left-shares of $[[\mathbf{A}]]_i^L$, enumerated as $[[\mathbf{A}]]_{i,j}^{LL}=\big[\big[[[\mathbf{A}]]_i^L\big]\big]_j^L, \forall j\in[N]$.*

4.6.3 Transpose of a matrix

Consider that server i stores a (N,K_1,T_1) left-share of $\mathbf{A}, i\in[N]$. The goal is to obtain a procedure, through which the servers end up with the (N,K_2,T_2) left-shares of $\mathbf{A}^{tr}$ instead. The procedure takes the following steps:

1. Server $i\in[N]$ performs the transpose operation on the left-share $[[\mathbf{A}]]_i^L$, to obtain a share $[[\mathbf{A}^{tr}]]_i=([[\mathbf{A}]]_i^L)^{tr}$. We have

$$\begin{aligned}
[[\mathbf{A}^{tr}]]_i&=([[\mathbf{A}]]_i^L)^{tr} && (4.55)\\
&=\sum_{l=1}^{K_1}\mathbf{A}_l^{tr}\alpha_N^{(i-1)(l-1)}+\sum_{l=1}^{T_1}\mathbf{R}_l^{tr}\alpha_N^{(i-1)(K+l-1)}, && (4.56)
\end{aligned}$$

where $\mathbf{A}_l^{tr}, l = 1, \ldots, K_1$ are equivalent to row-wise partitions of $\mathbf{A}^{tr}$. Note that $[[\mathbf{A}^{tr}]]_i$ is neither a left-share nor a right-share, but is equivalent to having a row-wise partitioning of $\mathbf{A}^{tr}$ employed in right-encoding, and having the exponents of the secret keys employed in left-encoding.

2. Server i then generates (N, K_2, T_2) left-shares of $[[\mathbf{A}^{tr}]]_i$ evaluated on the N^{th} roots of unity $\alpha_N^{j-1}, \forall j \in [N]$, and enumerated as $[[\mathbf{A}^{tr}]]_{i,j}^L = \left[\left[[[\mathbf{A}^{tr}]]_i\right]\right]_j^L, \forall j \in [N]$.

3. Server i sends the left-share $[[\mathbf{A}^{tr}]]_{i,j}^L$ to server j.

4. Server j interpolates the received shares $[[\mathbf{A}^{tr}]]_{i,j}^L, i = 1, \ldots, N$, using IFFT to obtain the first K_1 coefficients, which are K_1 row-wise partitions of the share $[[\mathbf{A}^{tr}]]_j^L$. Stacking them row-wise gives the share $[[\mathbf{A}^{tr}]]_j^L$.

4.6.4 Exponentiation

Suppose the user wants to compute $G(\mathbf{A}) = \mathbf{A}^r, r \in \mathbb{N}$. $(N, N-2T, T)$ left-shares of matrix $\mathbf{A}$ are sent to the servers. The servers then implement the scheme to generate $(N, N-2T, T)$ right-shares of matrix $\mathbf{A}$ as described in Section 4.6.2. Thus, the servers now have the shares of matrix $\mathbf{A}$ for left and right multiplication. The servers then implement the scheme for multiple matrix multiplication described in Section 4.5. If $r = 2^n$, the computation can be done in $\log r$ rounds of computation and communication phases by computing $\mathbf{A}^2, \mathbf{A}^4, \mathbf{A}^8, \ldots, \mathbf{A}^r$ in successive rounds. If $r \neq 2^n$, then consider the binary expansion $\{a_{B-1}, \ldots, a_0\}$ of r, such that $r = \sum_{i=0}^{B-1} a_i 2^i$, where B is the maximum number of bits required to represent r. Then, the desired result is given by

$$\mathbf{A}^r = \mathbf{A}^{\sum_{i=0}^{B-1} a_i 2^i} \tag{4.57}$$

$$= \prod_{i=0}^{B-1} \mathbf{A}^{a_i 2^i}. \tag{4.58}$$

If the Hamming weight of the binary expansion of r is h, then the computation requires $h-1$ extra rounds of computation and communication phases to perform the computation of Eq. (4.58), resulting in a total of $\log r + h - 1$ rounds.

Upload cost: $(N, N-2T, T)$ left-shares of matrix $\mathbf{A}$ are uploaded by the source, while the $(N, N-2T, T)$ right-shares of matrix $\mathbf{A}$ are generated in-situ by the share-conversion algorithm, therefore not required to be sent by the source. Therefore, we have $\chi_{UL} = \frac{N}{N-2T}$.

4.6.5 Solving the linear system $\mathbf{AX} = \mathbf{B}$ with secure Gaussian elimination

The linear system $\mathbf{AX} = \mathbf{B}$, where the elements of $\mathbf{A}$ and $\mathbf{B}$ belong to $\mathbb{F}_q$, can be solved by performing Gaussian elimination (GE) on the augmented matrix $(\mathbf{A}|\mathbf{B})$. Setting $\mathbf{B}$ equal to the identity matrix, the solution of the linear system also gives the matrix inverse $\mathbf{A}^{-1}$, if it exists. The GE method performs elementary row operations on the augmented matrix of the linear system, and row interchanges, also called pivoting, to transform the linear system into its row-echelon form. A scheme for secure GE is described in [92], which takes element-wise secret shares of matrices $\mathbf{A}$ and $\mathbf{B}$ as inputs, and outputs the solution of the linear system. Computing element-wise secret shares of a matrix is equivalent to computing $(N, 1, T)$ secret shares of matrices $\mathbf{A}$ and $\mathbf{B}$. Therefore, if the servers store (N, K, T) secret shares of matrix $\mathbf{A}$ and $\mathbf{B}$, they must first be converted to $(N, 1, T)$ secret shares using the scheme in Section 4.6.2.

4.6.6 Secure matrix inversion

Besides solving the linear system $\mathbf{AX} = \mathbf{I}$ to compute the matrix inverse, as described in the previous section, a different procedure, inspired from that in [93], is described below, through which the servers start from (N, K, T) right-shares of a square matrix $\mathbf{A} \in \mathbb{F}_q^{m \times m}$, and end up with (N, K, T) left-shares of $\mathbf{A}^{-1}$ instead.

1. A uniformly distributed random matrix $\mathbf{\Phi} \in \mathbb{F}_q^{m \times m}$, which is used as the secret key, is secretly shared with the servers. We assume that there is no central entity that can generate and share the secret key securely with the servers. The shares of the secret key are generated by the servers in a decentralized manner as follows: For $i \in [N]$, server i generates a random matrix $\mathbf{\Phi}^{(i)} \in \mathbb{F}_q^{m \times m}$, and generates its

(N, K, T) left-shares evaluated on the N^{th} roots of unity $\alpha_N^{j-1}, \forall j \in [N]$. Server i sends the left-share $[[\mathbf{\Phi}^{(i)}]]_j^L$ to server $j \in [N] \setminus \{i\}$. From the left-shares received by server j, it computes the left-share $[[\mathbf{\Phi}]]_j^L = [[\mathbf{\Phi}^{(1)} + \cdots + \mathbf{\Phi}^{(N)}]]_j^L = [[\mathbf{\Phi}^{(1)}]]_j^L + \cdots + [[\mathbf{\Phi}^{(N)}]]_j^L$. Thus, each server obtains a left-share of a common secret key $\mathbf{\Phi} = \mathbf{\Phi}^{(1)} + \cdots + \mathbf{\Phi}^{(N)}$.

2. Server j securely computes $[[\mathbf{P}]]_j = [[\mathbf{\Phi A}]]_j = [[\mathbf{\Phi}]]_j^L [[\mathbf{A}]]_j^R$.

3. The servers reconstruct matrix $\mathbf{P}$ from its secret shares by exchanging their secret shares of matrix $\mathbf{P}$ with every other server, and averaging the received shares, similar to Eq. (4.19). Thus, each server obtains the public matrix $\mathbf{P} = \mathbf{\Phi A}$. The servers gain no information of the matrix $\mathbf{A}$ from the matrix $\mathbf{P}$, therefore satisfying the privacy constraint against the servers.

4. Each server computes the matrix inverse $\mathbf{P}^{-1} = \mathbf{A}^{-1}\mathbf{\Phi}^{-1}$.

5. For $j = 1, \ldots, N$, server j then obtains a left-share of the inverse of matrix $\mathbf{A}$ as follows: $[[\mathbf{A}^{-1}]]_j^L = \mathbf{P}^{-1}[[\mathbf{\Phi}]]_j^L = [[\mathbf{P}^{-1}\mathbf{\Phi}]]_j^L$.

6. The servers can now perform further computation on the left-shares obtained, or deliver their left-shares to the user, who then performs IFFT on the received results to obtain the matrix $\mathbf{A}^{-1}$.

Upload cost: $(N, N-2T, T)$ shares of matrix $\mathbf{A}$ are uploaded by the source for Step 2, where SDMM of random matrix $\mathbf{\Phi}$ and input matrix $\mathbf{A}$ is performed, thus incurring an upload cost of $\chi_{UL} = \frac{N}{N-2T}$.

4.6.7 Iterative matrix inversion

The method for secure matrix inversion introduced above includes an intermediate step (step 4) that involves the inversion of a secure full-size matrix at each server. While the procedure satisfies the privacy requirements, it may contradict with the motivation of distributed computation. Iterative matrix inversion algorithms, for example Newton's method [98], do not involve direct matrix inversions, but instead proceed

with matrix multiplications, and therefore, are amenable to efficient distributed implementation. Newton's method, however, provides only an approximation of the matrix inverse. Newton's method for inverting matrices is derived from Newton's method for finding the root of a function. The procedure is as follows [98]:

- Set $f(\mathbf{X}) = \mathbf{A} - \mathbf{X}^{-1}$. Note that the root of $f(\mathbf{X})$ is $\mathbf{A}^{-1}$. Apply Newton's method for finding its root, as follows:

$$\mathbf{X}_{i+1} = \mu_i \mathbf{X}_i \left(2\mathbf{I} - \mathbf{A}\mathbf{X}_i\right), \tag{4.59}$$

  where $\mu_i = 1$ for all $i > 0$.

- **Choosing the initial estimate $\mathbf{X}_0$:** Quadratic convergence is obtained if $||\mathbf{A}\mathbf{X}_0 - \mathbf{I}|| < 1$. This is satisfied if $\mathbf{X}_0 = \mu_o \mathbf{A}^T$ is picked as the initial estimate, with the value of μ_0 as proposed in [98].

The matrix addition and multiplication operations can be performed securely in a distributed manner using the algorithms described in this chapter.

4.6.8 Computation of arbitrary polynomials:

The algorithms for securely performing matrix addition, transpose, exponentiation, inverse and multiplication that have been described in this chapter allow the user to compute arbitrary matrix polynomials on distributed servers. Therefore, a function of the following form,

$$G(\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3) = \mathbf{A}_1^2 \mathbf{A}_2 + c\mathbf{A}_3^{-1}, \tag{4.60}$$

for example, can be computed securely on distributed servers as follows: first, the servers securely compute $\mathbf{A}_1^2\mathbf{A}_2$ using the scheme for multiple matrix multiplication; then, the servers add the shares $[[\mathbf{A}_1^2\mathbf{A}_2]]$ to the shares $[[c\mathbf{A}_3^{-1}]]$, computed using one of the secure matrix inversion methods described, to finally obtain the secret shares of the final result, which the user receives and decodes to obtain the desired result.

4.6.9 Secure learning from local datasets

Consider that data from D source nodes, each with a different size of dataset, is used for training a fully connected deep neural network. In a fully connected neural network, the input layer of neurons performs the matrix multiplication $\mathbf{WX}$, where $\mathbf{W}$ is the weight matrix associated with the layer of neurons, and $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_D)$, where $\mathbf{X}_i, i \in [D]$ is the dataset belonging to source node i. SDMM schemes in [62,64,70,95], based on column-wise partitioning of matrix $\mathbf{X}$, code across different data points, thus requiring the local datasets of the sources to be encoded at a central location, which leads to a privacy concern. In contrast, the SDMM algorithm proposed in this chapter, based on row-wise partitioning of matrix $\mathbf{X}$, encodes each dataset independently. We have

$$[[\mathbf{X}]]^R = \left([[\mathbf{X}_1]]^R, \ldots, [[\mathbf{X}_D]]^R\right), \tag{4.61}$$

that is, each source can deliver the right shares of its dataset to the servers independent of other sources.

Linear regression- Consider the computation of the MMSE estimate in a linear regression problem, where the optimum estimate is given by $\beta = \left(\mathbf{X}^{tr}\mathbf{X}\right)^{-1}\mathbf{X}^{tr}\mathbf{Y}$. The local datasets $(\mathbf{X}_i, \mathbf{Y}_i)$ can be delivered to the servers similarly to Eq. (4.61), and then the left-shares and right-shares of $\mathbf{X}^{tr}$ and $\mathbf{Y}$ can be generated by using the algorithms to convert shares and perform secure transpose described in this section.

4.6.10 Achieving the optimal upload and download cost for SDMM

If we assume that the cost of inter-server communication is negligible compared to the upload and download costs, optimal communication costs can be achieved simultaneously for both upload and download while ignoring the inter-server communication costs. The source nodes upload $(N, N-T, T)$ shares of the input matrices to the servers, resulting in an upload cost of $\chi_{UL} = \frac{N}{N-T}$, which is shown to be a lower bound on the upload cost for SDMM in [95]. Using the algorithm proposed in Section 4.6.2, the $(N, N-T, T)$ shares can then be converted to $(N, N-2T, T)$ shares for

implementing the SDMM algorithm proposed in this chapter. The secret shares of the computation results are then converted to $(N, N-T, T)$ shares using the algorithm in Section 4.6.2. These shares are sent to the user, resulting in a download cost of $\chi_{DL} = \frac{N}{N-T}$, which is known to be the optimal download cost for SDMM [62]. This procedure circumvents the trade-off between the upload cost and the download cost for SDMM schemes, studied earlier in [95].

The assumption that the cost of inter-server communication is negligible is justified in many practical scenarios involving computing clusters, where the computing servers are connected with high-speed communication links, while the links between the source nodes and the servers, and between the user and the servers may have limited bandwidth. However, when the inter-server communication costs (delay, bandwidth and/or energy) are non-negligible, the extra rounds of communication among the servers, required for various share conversions, become prohibitive.

4.7 SDMC in Integer Rings

We discussed in section 4.1.3 that arithmetic on real numbers can be done by rounding them off to the closest rational number of fixed precision, and encoding them into a finite field. Finite fields are quotient rings with a prime characteristic. In this section, we generalize the proposed SDMC schemes to rings with a non-prime characteristic.

Consider that the elements of the input matrices $\mathbf{A}^{(\gamma)}, \gamma \in [\Gamma]$, lie in the integer ring $\mathbb{Z}/n\mathbb{Z}, n \in \mathbb{N}$. Let z_N be a primitive N^{th} root of unity in $\mathbb{Z}/n\mathbb{Z}$. To compute, for example, the left-shares of the matrices $\mathbf{A}^{\gamma}, \gamma = 1, \ldots, \Gamma$, the proposed scheme evaluates the polynomials $\mathbf{A}^{(\gamma)}(x), \gamma = 1, \ldots, \Gamma$, at the N^{th} roots of unity $1, z_N, z_N^2, \ldots, z_N^{N-1}$. The encoding (and decoding) scheme is equivalent to computing the N-point discrete Fourier transform (and inverse discrete Fourier transform) of the sequence $\left\{\mathbf{A}_1^{(\gamma)}, \ldots, \mathbf{A}_K^{(\gamma)}, \mathbf{R}_1^{(\gamma)}, \ldots, \mathbf{R}_T^{(\gamma)}\right\}, \gamma = 1, \ldots, \Gamma$. The theory of discrete Fourier transform works correctly if and only if z_N is a principal root of unity, that is, if it

satisfies the following condition:

$$\sum_{i=0}^{N-1} z_N^{is} = 0, \quad \forall s \in [N-1]. \tag{4.62}$$

In general, not every primitive root of unity in $\mathbb{Z}/n\mathbb{Z}$ is a principal root of unity. Therefore, we state the following theorems relating to the existence and construction of principal roots of unity in $\mathbb{Z}/n\mathbb{Z}$.

Remark 4.6. *The correct working of the inverse Fourier transform requires the inverse of N to exist in the integer ring, since division by N is required. However, in the SDMC scheme we assume that the integer ring is much larger than the final result, which is necessary for computation on data in rings to reflect computation on raw integer data, as described in section 4.1.3. Therefore, the result can be lifted to $\mathbb{Z}$, where division by N can be done with conventional arithmetic. The trick is in the fact that lifting and dividing works if the value does not wrap around the ring modulus.*

4.7.1 Finding the principal N^{th} roots of unity

Theorem 4.4. *Let $n = p^k$, where p is prime. For $N < p$, every primitive N^{th} root of unity is a principal N^{th} root of unity.*

Proof. Let z_N be a primitive N^{th} root of unity in $\mathbb{Z}/p^k\mathbb{Z}$. The identity $\sum_{j=0}^{N-1} z_N^j = 0$ always holds if $z_N - 1$ is not a zero divisor. Moreover, z_N is a principal root of unity if $z_N^l - 1$ is not a zero divisor for any $l \in [N-1]$.

Assume that $z_N^l - 1$ is a zero divisor for some $l \in \mathbb{Z}$. Then, $\gcd(z_N^l - 1, p^k) = p^h, h < k$. In other words, we can write $z_N^l - 1 = sp^h$ for some $s \in \mathbb{Z}$. Then, we have

$$(z_N^l)^N - 1 = (z_N^l - 1)\left(1 + z_N^l + \cdots + (z_N^l)^{N-1}\right) \tag{4.63}$$

$$= (z_N^l - 1)\left(1 + (sp^h + 1) + \cdots + (sp^h + 1)^{N-1}\right) \tag{4.64}$$

$$= sp^h \left(N + p^h \sum_{j=0}^{N-2} a_j p^{hi}\right). \tag{4.65}$$

If $N < p$, then the second term is coprime to p. Therefore, the product of these two terms cannot be a multiple of p^k, thus not equal to zero mod p^k, which is a contradiction to our priori assumption that z_N^l is an N^{th} root of unity. Therefore, $z_N^l - 1$ cannot be a zero divisor for any $l \in [N-1]$ if $N < p$, implying that z_N is a principal root of unity. □

Theorem 4.5. *Let $n = m_1 m_2, m_1, m_2 \in \mathbb{Z}$, where m_1 and m_2 are co-prime. If there exists a principal N^{th} root of unity in $\mathbb{Z}/m_1\mathbb{Z}$, then there exists at least one principal N^{th} root of unity in $\mathbb{Z}/n\mathbb{Z}$ if $m_2|N$.*

Proof. Consider the integer ring $\mathbb{Z}/n\mathbb{Z} \cong (\mathbb{Z}/m_1\mathbb{Z}) \times (\mathbb{Z}/m_2\mathbb{Z})$. Consider a principal N^{th} root of unity z_N in $\mathbb{Z}/m_1\mathbb{Z}$. Consider G, a cyclic subgroup of $(\mathbb{Z}/n\mathbb{Z})^\times \cong (\mathbb{Z}/m_1\mathbb{Z})^\times \times (\mathbb{Z}/m_2\mathbb{Z})^\times$, which is given by $G = \{(z_N^i, 1) : i = 0, 1, \ldots, N-1\}$. Then, for $s = 1, 2, \ldots, N-1$, we have

$$\sum_{i=0}^{N-1} (z_N^{is}, 1) = \left(\sum_{i=0}^{N-1} z_N^{is}, \sum_{i=0}^{N-1} 1 \right) \tag{4.66}$$

$$= (0, 0) \quad \text{if } m_2|N. \tag{4.67}$$

Therefore, the generator of the group G is a principal N^{th} root of unity. □

Theorem 4.6. *Let $n = m_1 m_2, m_1, m_2 \in \mathbb{Z}$, where m_1 and m_2 are co-prime. If there exists a principal N^{th} root of unity in both $\mathbb{Z}/m_1\mathbb{Z}$ and $\mathbb{Z}/m_2\mathbb{Z}$, then there exists at least one principal N^{th} root of unity in $\mathbb{Z}/n\mathbb{Z}$.*

Proof. Consider the integer ring $\mathbb{Z}/n\mathbb{Z} \cong (\mathbb{Z}/m_1\mathbb{Z}) \times (\mathbb{Z}/m_2\mathbb{Z})$. Consider a principal N^{th} root of unity y_N in $\mathbb{Z}/m_1\mathbb{Z}$, and a principal N^{th} root of unity z_N in $\mathbb{Z}/m_2\mathbb{Z}$. Consider G, a cyclic subgroup of $(\mathbb{Z}/n\mathbb{Z})^\times \cong (\mathbb{Z}/m_1\mathbb{Z})^\times \times (\mathbb{Z}/m_2\mathbb{Z})^\times$, which is given by $G = \{(y_N^i, z_N^i) : i = 0, 1, \ldots, N-1\}$. Then, for $s = 1, 2, \ldots, N-1$, we have

$$\sum_{i=0}^{N-1} (y_N^{is}, z_N^{is}) = \left(\sum_{i=0}^{N-1} y_N^{is}, \sum_{i=0}^{N-1} z_N^{is} \right) \tag{4.68}$$

$$= (0, 0). \tag{4.69}$$

Therefore, the generator of the group G is a principal N^{th} root of unity. □

4.7.2 Examples

1. *Prime powers:* For $n = 11^3$, by applying Theorem 4.4, we obtain $z_{10} = 596$ as a principal 10^{th} root of unity, which generates the group of roots of unity $(1, 596, 1170, 1207, 632, 1330, 735, 161, 124, 699)$. Similarly, $z_5 = 1170$ is a principal 5^{th} root of unity mod n, and generates the group $(1, 1170, 632, 735, 124)$.

2. *Other composite numbers:* For $n = 6 * 19^2$, by applying Theorem 4.5, we obtain $z_{18} = 1921$ as a principal 18^{th} root of unity, and $z_6 = 1015$ as a principal 6^{th} root of unity.

3. Let $n = 13^2 * 19^2$. Since there exist principal 6^{th} roots of unity in both $\mathbb{Z}/13^2\mathbb{Z}$ and $\mathbb{Z}/19^2\mathbb{Z}$ (from Theorem 4.4), there also exists a principal 6^{th} root of unity in $\mathbb{Z}/n\mathbb{Z}$, due to Theorem 4.6. Indeed, we obtain $z_6 = 9318$ as a principal 6^{th} root of unity in $\mathbb{Z}/n\mathbb{Z}$, which generates the roots of unity $\{1, 9318, 9317, 61008, 51691, 51692\}$.

4. Let $n = (2 * 7) * 13^2$. Since there exists a 6^{th} principal root of unity in $\mathbb{Z}/14\mathbb{Z}$, due to Theorem 4.5, and there exists a principal 6^{th} root of unity in $\mathbb{Z}/13^2\mathbb{Z}$, due to Theorem 4.4, we obtain a principal 6^{th} root of unity $z_6 = 2175$ in $\mathbb{Z}/n\mathbb{Z}$, due to Theorem 4.6.

4.8 SDMM in Integer Rings

In this section we briefly demonstrate how SDMM can be performed in an integer ring. We first consider the case when N is coprime to the modulus n of the ring, which admits a multiplicative inverse of N, thus allowing the computation of the IFFT. In this case, the SDMM encoding and decoding scheme is identical to that described in Section 4.3, where the evaluation points of the matrix polynomials are chosen to be the N^{th} principal roots of unity in $\mathbb{Z}/n\mathbb{Z}$, if they exist. We consider the case when N is not coprime to n in the following subsection.

4.8.1 N not coprime to n

To multiply matrices $\mathbf{A}$ and $\mathbf{B}$, their secret shares $[[\mathbf{A}]]_i^L$ and $[[\mathbf{B}]]_i^R$, $i = 1, \ldots, N$ are generated, similarly to Section 4.3. Server i computes $[[\mathbf{C}]]_i = [[\mathbf{A}]]_i^L [[\mathbf{B}]]_i^R$, and delivers it to the user. Ideally the user would decode the result by computing the average of the received results same as in Eq. (4.19), but this cannot be computed since the multiplicative inverse of N does not exist in the ring. Instead, using Bezout's identity, we have

$$aN = bn + \gcd(N, n), \tag{4.70}$$

for some constants $a, b \in \mathbb{Z}$. Therefore, the user computes the following modification of Eq. (4.19):

$$a \sum_{i=1}^{N} [[\mathbf{C}]]_i = \gcd(N, n) \sum_{l=1}^{K} \mathbf{A}_l \mathbf{B}_l. \tag{4.71}$$

Assuming that the right-hand side term does not wrap around the modulus of the ring, the user can finally obtain the result $\sum_{l=1}^{K} \mathbf{A}_l \mathbf{B}_l$ by lifting the right-hand side term to $\mathbb{Z}$ and then dividing by $\gcd(N, n)$.

This method can also be applied to SDMM of multiple matrices, that is, for $\Gamma > 2$. Say matrices $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ have to be multiplied. In the first round, the constant $\gcd(N, n)$ is obtained as a multiplicative factor to the result $\mathbf{C}^{(2)}$. In the second round, a further multiplicative factor of $\gcd(N, n)$ is obtained, thus giving the factor $\gcd(N, n)^2$, which can again be reduced to the factor $\gcd(N, n)$ by using Bezout's identity. In this way, the multiplicative factor is prevented from growing even if multiple rounds of secure computation are performed.

4.8.2 Comparison with existing schemes

Secure distributed matrix computation schemes in existing literature are mostly based on polynomial interpolation, often implemented using the lagrange interpolation method. Polynomial interpolation in arbitrary integer rings is not always feasible unless

all the divided differences of the coordinates being interpolated exist, that is, the values $(x_i - x_j), i \neq j$, where x_i and x_j are distinct evaluation points used for generating the secret shares, have a multiplicative inverse in $\mathbb{Z}/n\mathbb{Z}$. The scheme proposed in this chapter, which is based on a rather simple decoding process using the division-free inverse discrete Fourier transform algorithm, mitigates the problem of the existence of multiplicative inverses as long as suitable principal roots of unity can be found.

4.9 Discussion and Conclusions

In this chapter we developed a novel polynomial coded computation scheme achieving a near-optimal performance in terms of the upload cost for SDMM across N servers, any T of which may collude. We also proposed a scheme achieving the optimal upload cost for the special case when the user requesting the computation is also the source of the matrices to be computed upon. The scheme involves evaluating the constructed polynomials at the roots of unity in an appropriate finite field, which is equivalent to taking the discrete Fourier transform of the constituent matrices. The encoding and decoding complexity is also lower than all the other schemes in the literature. For a special case of the data matrices having certain asymptotic dimensions, our scheme also achieves the optimal download cost. We also introduced a method for straggler mitigation, which provides group-wise tolerance to straggling servers. Straggler tolerance is achieved at the expense of an increase in the upload cost. We further extended our scheme to implement multiplication of multiple matrices, while keeping the input matrices and all the intermediate computations secure against any T colluding servers, with a minimal upload cost. This presents a substantial improvement in performance in terms of the upload cost for multiplication of multiple matrices over existing schemes in the literature. Moreover, we described procedures for other common matrix operations, some of which can be reduced to a set of matrix multiplications, thus allowing us to compute arbitrary matrix polynomials. We also generalized the proposed schemes to arbitrary integer rings, a characteristic which is unique to our scheme when compared with the schemes in existing literature.

Resources

An implementation in SageMath of the algorithms for secure matrix computations described in this chapter is available on Github.

Chapter 5

Conclusions

Storage and computing resources are integral to information-centric communication networks of the present and the future, and with drastically increasing data traffic on the internet, efficient techniques of storage and computing are necessary. There is a growing trend of pushing the data storage and computing resources to the absolute edge of the network, closer to the end user and the source of the data. As a result, the density of storage (or caching) servers and computing servers on the edge of the network is projected to grow exponentially. When storage and compute functions are close to the end user, ultra-fast, low latency data retrieval and computation are achievable, which consequentially gives way to new applications like cloud gaming, augmented reality (AR), virtual reality (VR), federated neural network learning, etc.

This thesis can be divided into three parts, each tackling a different aspect of network operation. In Chapter 2, the role of coded caching is explored in a distributed edge network, where small base stations (SBSs) proactively cache popular content, and deliver the content to cache-enabled end users on demand. Major challenges in edge caching are to guarantee that the most frequently accessed content is available on the edge servers (higher cache-hit ratio) while using efficient storage schemes, to discover the location of requested contents in the edge caches, and to efficiently deliver the content to the end user. These challenges are addressed in Chapter 2 by distributing the content across multiple storage-constrained servers with coded redundancy, so that the end user can pull the content from a threshold number of the nearest cache servers. This increases the probability of the end user finding nearby caches storing the desired content. Moreover, contents are multicasted simultaneously to multiple cache-enabled end users, possibly with different requests, over a shared channel, using novel coding techniques which drastically reduce the bandwidth requirement of the network. Chapter 2 highlights multiple trade-offs that play a role in caching networks, namely, the

trade-off between the communication bandwidth and the end-user cache capacity, as well as the trade-off between the communication bandwidth and the server storage capacity.

In Chapter 3, aspects related to distributed storage (or caching) systems are studied in greater detail, modeling the network of SBS caches as a distributed storage system. The main problem studied in Chapter 3 is that of performing efficient repair of failed storage nodes while providing efficient storage. In the context of distributed SBS caches considered in Chapter 2, the distributed storage repair problem models the scenario where the SBS caches which are accessible to the end user do not store the required content; hence, instead of employing the backhaul to fetch the content from the origin servers to the SBS caches that are accessible to the end user, the SBS caches use their interconnected network to "repair" new SBS cache nodes, or in other words, move content to new SBS caches which are accessible to the end user. Chapter 3 studies the trade-off between the repair bandwidth and the storage capacity of each cache node, and constructs a storage and repair framework which is optimal in many performance metrics. Similarly to Chapter 2, the interplay between storage and communication is highlighted.

In Chapter 4, the interplay between multiple aspects of network operation are studied together, namely, between computation, storage, communication, and privacy. In Chapter 4, secure multiparty computation is studied, where a user wants to offload a computation on massive matrices, possibly produced by third party sources, to a group of N edge servers, while guaranteeing that the data is kept private from the computing servers, and even the user learns nothing about the original data beyond the computation result. This is an especially interesting area of research today due to its implications for large scale private machine learning. This is an absolutely necessary feature in future communication networks and devices for the democratization of research and innovation.

This thesis highlights multiple applications of coding theory in communication networks, beyond the conventional applications for communication over noisy channels. Adoption of these techniques on a large scale can provide significant benefits, but there are still many challenges relating to technical aspects as well as standardization. It

is stated in [2] that: "Employing caching in wireless networks has been discussed and proposed by many companies in the scope of 3GPP. For example, T-DOC R3-160688 proposes to place an edge cache at an LTE base station either embedded in the eNodeB or standalone. T-DOC R3-160828 explores the benefits of local caching. In the scope of 5G-RAN R.14, the report 3GPP TR 36.933 describes the different caching modules that are included in 5G base stations". The standardization efforts pave the road for further research in wireless caching, and introducing coding techniques in the standards will go a long way in guaranteeing seamless operation of communication networks.

Meanwhile, standardization and regulatory efforts for user data privacy are not lacking. For example, the report NISTIR 8214A puts forward a preliminary roadmap towards the standardization of threshold schemes for cryptographic primitives. The standardization process also includes soliciting feedback from a broad audience, to aid the development of the roadmap for standardization. The onus is on the open source communities and academic researchers to develop semantically or information-theoretically secure systems, and prove their feasibility to be included in the standards.

5.0.1 Research Challenges

Besides the standardization efforts, significant effort is required to develop techniques for robust adoption on a large scale. Coded caching techniques suffer from the subpacketization problem, which limits the number of users that can be served using coded caching, thus limiting the coding gains that can be achieved. Recent research efforts have managed to trade-off caching gains with subpacketization, and suggest novel methods such as introducing multiple antennas at the transmitter to decrease the level of subpacketization. The scheme proposed in Chapter 2 can also be adapted to low subpacketization schemes, like the Placement Delivery Array (PDA) based scheme proposed in [80]. Another interesting open problem for future work is finding a lower bound and an optimal scheme when fractional repetition codes are used for distributed storage in SBSs as studied in Chapter 2. A toy example (Fig. 2.6) in Chapter 2 illustrates the potential benefits of FR codes in coded caching problems. The toy example

presents an asymmetry in the user connections to the servers, where users 1 and 3 connect to one server each, while user 2 connects to 2 servers. An interesting problem is constructing a general scheme for heterogeneous network topologies and extracting gains from such topologies as demonstrated in the toy example. Another question relates to gains from heterogeneous distributed storage. For instance, if there is knowledge of user dynamics and non-uniform probabilities of the user-server connections, can a heterogeneous distributed storage scheme be designed to extract higher average gains through coded caching? An extreme case of this scenario would mimic the combination network model, where the user-server connections are completely fixed and known, and where coded caching achieves higher gains. Such open problems present ripe material for future research.

Private distributed computation is a relatively young research topic, and while some open source distributions exist that implement homomorphic encryption and secure multiparty computation (SMPC) schemes, the computation speeds are still very slow, and require a large amount of communication among the computing parties, especially for non-linear computations. Microsoft SEAL [99] is an open source implementation of homomorphic encryption schemes, while Crypten by Facebook and Openmined PySyft [100] are open-source libraries that implement distributed neural network training and inference using secure multiparty computation. Computation speeds may be increased significantly by reducing the computation complexity of primitives like integer and matrix multiplication in finite rings. Also, while SMPC schemes provide a lightweight alternative to homomorphic encryption based schemes, SMPC algorithms for non-linear computations require a large number of communication rounds among the computing servers, and reducing the round complexity of non-linear secure multiparty computations is an important direction of research. Future work that builds on the schemes described in Chapter 4 may include methods for securely performing matrix decompositions and other non-linear matrix computations on distributed servers. It would also be interesting to develop schemes for multiplication of an arbitrary number of matrices with minimum inter-server communication.

In conclusion, we addressed multiple challenges in emerging edge networks, which demonstrate the gains provided by coding techniques in achieving optimal trade-offs

between communication, storage, computing, and security/privacy. It is highly likely that such coding techniques will provide efficient solutions to many other problems, such as topics in network function virtualization (NFV) and software defined networking (SDN), and we hope that the research results presented in this thesis have contributed to the development of efficient communication networks, as well as to further encourage research and development in this area so that remaining open issues will be solved and the full potential of the future communication networks will be unleashed.

Bibliography

[1] N.˜Golrezaei, A.˜Molisch, A.˜Dimakis, and G.˜Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Comm. Magazine*, vol.˜51, no.˜4, 2013.

[2] G.˜S. Paschos, G.˜Iosifidis, M.˜Tao, D.˜Towsley, and G.˜Caire, "The role of caching in future communication systems and networks," *IEEE Journal on Selected Areas in Communications*, vol.˜36, no.˜6, pp. 1111–1125, 2018.

[3] M.˜A. Maddah-Ali and U.˜Niesen, "Fundamental limits of caching," *IEEE Trans. on Inf. Theory*, vol.˜60, no.˜5, pp. 2856–2867, May 2014.

[4] N.˜Mital, D.˜Gündüz, and C.˜Ling, "Coded caching in a multi-server system with random topology," in *IEEE Wireless Communications and Networking Conference (WCNC)*, April 2018, pp. 1–6.

[5] M.˜Mohammadi Amiri, Q.˜Yang, and D.˜Gündüz, "Decentralized caching and coded delivery with distinct cache capacities," *IEEE Transactions on Communications*, vol.˜65, no.˜11, pp. 4657–4669, 2017.

[6] K.˜Shanmugam, N.˜Golrezaei, A.˜G. Dimakis, A.˜F. Molisch, and G.˜Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol.˜59, no.˜12, pp. 8402–8413, 2013.

[7] M.˜Franceschetti and R.˜Meester, *Random Networks for Communication: From Statistical Physics to Information Systems*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2008.

[8] M.˜Mohammadi Amiri and D.˜Gündüz, "Fundamental limits of coded caching: Improved delivery rate-cache capacity tradeoff," *IEEE Transactions on Communications*, vol.˜65, no.˜2, pp. 806–815, 2017.

[9] M.˜Gregori, J.˜Gómez-Vilardebó, J.˜Matamoros, and D.˜Gündüz, "Wireless content caching for small cell and d2d networks," *IEEE Journal on Selected Areas in Comm.*, vol.˜34, no.˜5, pp. 1222–1234, May 2016.

[10] J.˜Gómez-Vilardebó, "Fundamental limits of caching: Improved rate-memory tradeoff with coded prefetching," *IEEE Transactions on Communications*, vol.˜66, no.˜10, pp. 4488–4497, 2018.

[11] C.˜Tian and J.˜Chen, "Caching and delivery via interference elimination," *IEEE Transactions on Information Theory*, vol.˜64, no.˜3, pp. 1548–1560, 2018.

[12] T.˜Luo, V.˜Aggarwal, and B.˜Peleato, "Coded caching with distributed storage," *IEEE Transactions on Information Theory*, vol.˜65, no.˜12, pp. 7742–7755, 2019.

[13] S.˜P. Shariatpanahi, S.˜A. Motahari, and B.˜H. Khalaj, "Multi-server coded caching," *IEEE Transactions on Information Theory*, vol.˜62, no.˜12, pp. 7253–7271, 2016.

[14] M.˜Ji, M.˜F. Wong, A.˜M. Tulino, J.˜Llorca, G.˜Caire, M.˜Effros, and M.˜Langberg, "On the fundamental limits of caching in combination networks," in *2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2015, pp. 695–699.

[15] A.˜G. Dimakis, P.˜B. Godfrey, Y.˜Wu, M.˜J. Wainwright, and K.˜Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. on Information Theory*, vol.˜56, no.˜9, pp. 4539–4551, Sept 2010.

[16] A.˜A. Zewail and A.˜Yener, "Coded caching for combination networks with cache-aided relays," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2433–2437.

[17] E.˜Ozfatura and D.˜Gündüz, "Mobility and popularity-aware coded small-cell caching," *IEEE Comm. Letters*, vol.˜22, no.˜2, 2018.

[18] N.˜Mital, K.˜Kralevska, C.˜Ling, and D.˜Gündüz, "Storage-repair bandwidth trade-off for wireless caching with partial failure and broadcast repair," in *2018 IEEE Information Theory Workshop (ITW)*, Nov 2018, pp. 1–5.

[19] I.˜Tamo, Z.˜Wang, and J.˜Bruck, "Zigzag codes: Mds array codes with optimal rebuilding," *IEEE Transactions on Information Theory*, vol.˜59, no.˜3, pp. 1597–1616, March 2013.

[20] V.˜Guruswami and M.˜Wootters, "Repairing reed-solomon codes," in *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 216–226. [Online]. Available: https://doi.org/10.1145/2897518.2897525

[21] I.˜Tamo, M.˜Ye, and A.˜Barg, "Optimal repair of reed-solomon codes: Achieving the cut-set bound," *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 216–227, 2017.

[22] W.˜Li, Z.˜Wang, and H.˜Jafarkhani, "On the sub-packetization size and the repair bandwidth of reed-solomon codes," *IEEE Transactions on Information Theory*, vol.˜65, no.˜9, pp. 5484–5502, 2019.

[23] Z.˜Wang, I.˜Tamo, and J.˜Bruck, "Explicit minimum storage regenerating codes," *IEEE Transactions on Information Theory*, vol.˜62, no.˜8, pp. 4466–4480, 2016.

[24] A.˜E. Gamal and Y.-H. Kim, *Network Information Theory*. New York, NY, USA: Cambridge University Press, 2012.

[25] R.˜Ahlswede, Ning Cai, S.˜.˜R. Li, and R.˜W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol.˜46, no.˜4, pp. 1204–1216, July 2000.

[26] A.˜G. Dimakis and K.˜Ramchandran, "Deterministic regenerating codes for distributed storage yunnan," 2007.

[27] N.˜B. Shah, K.˜V. Rashmi, P.˜V. Kumar, and K.˜Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *2010 IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, 2010, pp. 1–5.

[28] K.˜V. Rashmi, N.˜B. Shah, and P.˜V. Kumar, "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, no.˜8, pp. 5227–5239, 2011.

[29] Y.˜Hu, Y.˜Xu, X.˜Wang, C.˜Zhan, and P.˜Li, "Cooperative recovery of distributed storage systems from multiple losses with network coding," *IEEE Journal Selected Areas in Comms.*, vol.˜28, no.˜2, 2010.

[30] A.˜M. Kermarrec, N.˜L. Scouarnec, and G.˜Straub, "Repairing multiple failures with coordinated and adaptive regenerating codes," in *2011 International Symposium on Networking Coding*, July 2011, pp. 1–6.

[31] K.˜W. Shum and Y.˜Hu, "Cooperative regenerating codes," *IEEE Trans. on Inf. Theory*, vol.˜59, no.˜11, pp. 7229–7258, Nov 2013.

[32] J.˜Chen and K.˜W. Shum, "Repairing multiple failures in the suh-ramchandran regenerating codes," in *2013 IEEE International Symposium on Information Theory*, 2013, pp. 1441–1445.

[33] K.˜Shum and J.˜Chen, "Cooperative repair of multiple node failures in distributed storage systems," *International Journal of Information and Coding Theory*, vol.˜3, 07 2016.

[34] M.˜Ye and A.˜Barg, "Cooperative repair: Constructions of optimal mds codes for all admissible parameters," *IEEE Transactions on Information Theory*, vol.˜65, no.˜3, pp. 1639–1656, 2019.

[35] Y.˜Zhang and Z.˜Zhang, "Scalar mscr codes via the product matrix construction," *IEEE Transactions on Information Theory*, vol.˜66, no.˜2, pp. 995–1006, 2020.

[36] A.˜Wang and Z.˜Zhang, "Exact cooperative regenerating codes with minimum-repair-bandwidth for distributed storage," in *2013 Proceedings IEEE INFOCOM*, April 2013, pp. 400–404.

[37] M.˜Gerami, M.˜Xiao, and M.˜Skoglund, "Partial repair for wireless caching networks with broadcast channels," *IEEE Wireless Communications Letters*, vol.˜4, no.˜2, pp. 145–148, April 2015.

[38] P.˜Hu, C.˜W. Sung, and T.˜H. Chan, "Broadcast repair for wireless distributed storage systems," in *Int. Conf. on Inf., Comms. and Signal Proc.*, Dec 2015, pp. 1–5.

[39] V.˜Abdrashitov, N.˜Prakash, and M.˜Medard, "The storage vs repair bandwidth trade-off for multiple failures in clustered storage networks," in *IEEE Inf. Theory Workshop (ITW)*, Nov 2017, pp. 46–50.

[40] A.˜S. Rawat, O.˜O. Koyluoglu, and S.˜Vishwanath, "Centralized repair of multiple node failures with applications to communication efficient secret sharing," *IEEE Transactions on Information Theory*, vol.˜64, no.˜12, pp. 7529–7550, Dec 2018.

[41] M.˜Zorgui and Z.˜Wang, "Centralized multi-node repair in distributed storage," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2016, pp. 617–624.

[42] ——, "Centralized multi-node repair regenerating codes," *CoRR*, vol. abs/1706.05431, 2017. [Online]. Available: http://arxiv.org/abs/1706.05431

[43] T.˜Ho, M.˜Medard, R.˜Koetter, D.˜R. Karger, M.˜Effros, J.˜Shi, and B.˜Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol.˜52, no.˜10, pp. 4413–4430, 2006.

[44] B.˜Shrader and A.˜Ephremides, "On packet lengths and overhead for random linear coding over the erasure channel," 2007.

[45] N.˜B. Shah, K.˜V. Rashmi, P.˜V. Kumar, and K.˜Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Transactions on Information Theory*, vol.˜58, no.˜3, pp. 1837–1852, March 2012.

[46] C.˜Tian, "Characterizing the rate region of the (4,3,3) exact-repair regenerating codes," *IEEE Journal on Selected Areas in Communications*, vol.˜32, no.˜5, pp. 967–975, May 2014.

[47] S.˜Ng and M.˜B. Paterson, "Functional repair codes: a view from projective geometry," *CoRR*, vol. abs/1809.08138, 2018. [Online]. Available: http://arxiv.org/abs/1809.08138

[48] H.˜D.˜L. Hollmann and W.˜Poh, "Characterizations and construction methods for linear functional-repair storage codes," in *2013 IEEE International Symposium on Information Theory*, July 2013, pp. 336–340.

[49] K.˜W. Shum and Y.˜Hu, "Functional-repair-by-transfer regenerating codes," in *2012 IEEE International Symposium on Information Theory Proceedings*, July 2012, pp. 1192–1196.

[50] S.˜Rane, W.˜Sun, and A.˜Vetro, "Secure function evaluation based on secret sharing and homomorphic encryption," in *Annual Allerton Conf. on Comm., Control, and Computing*, Sep. 2009, pp. 827–834.

[51] M.˜Yasuda, T.˜Shimoyama, J.˜Kogure, K.˜Yokoyama, and T.˜Koshiba, "Secure statistical analysis using rlwe-based homomorphic encryption," in *Information Security and Privacy*, E.˜Foo and D.˜Stebila, Eds. Cham: Springer International Publishing, 2015, pp. 471–487.

[52] D.˜H. Duong, P.˜K. Mishra, and M.˜Yasuda, "Efficient secure matrix multiplication over lwe-based homomorphic encryption," *Tatra Mountains Mathematical Publications*, vol.˜67, no.˜1, 2016.

[53] M.˜Yasuda, T.˜Shimoyama, J.˜Kogure, K.˜Yokoyama, and T.˜Koshiba, "Practical packing method in somewhat homomorphic encryption," in *Data Privacy Management and Autonomous Spontaneous Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 34–50.

[54] K.˜Lee, M.˜Lam, R.˜Pedarsani, D.˜Papailiopoulos, and K.˜Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol.˜64, no.˜3, pp. 1514–1529, March 2018.

[55] R.˜Tandon, Q.˜Lei, A.˜G. Dimakis, and N.˜Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine

Learning Research, D.˜Precup and Y.˜W. Teh, Eds., vol.˜70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 3368–3376. [Online]. Available: http://proceedings.mlr.press/v70/tandon17a.html

[56] Q.˜Yu, M.˜Maddah-Ali, and S.˜Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Advances in Neural Information Processing Systems 30*, I.˜Guyon, U.˜V. Luxburg, S.˜Bengio, H.˜Wallach, R.˜Fergus, S.˜Vishwanathan, and R.˜Garnett, Eds. Curran Associates, Inc., 2017, pp. 4403–4413.

[57] S.˜Dutta, M.˜Fahim, F.˜Haddadpour, H.˜Jeong, V.˜Cadambe, and P.˜Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol.˜66, no.˜1, pp. 278–301, Jan 2020.

[58] S.˜Dutta, Z.˜Bai, H.˜Jeong, T.˜M. Low, and P.˜Grover, "A unified coded deep neural network training strategy based on generalized polydot codes," *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1585–1589, 2018.

[59] Q.˜Yu, M.˜A. Maddah-Ali, and A.˜S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 2022–2026.

[60] Q.˜Yu, N.˜Raviv, J.˜So, and A.˜S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *CoRR*, vol. abs/1806.00939, 2018. [Online]. Available: http://arxiv.org/abs/1806.00939

[61] I.˜S. Reed and G.˜Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol.˜8, no.˜2, pp. 300–304, 1960. [Online]. Available: https://doi.org/10.1137/0108018

[62] W.-T. Chang and R.˜Tandon, "On the capacity of secure distributed matrix multiplication," *arXiv e-prints*, Jun. 2018.

[63] A.˜Shamir, "How to share a secret," *Commun. ACM*, vol.˜22, no.˜11, pp. 612–613, Nov. 1979.

[64] R.˜G.˜L. D'Oliveira, S.˜El Rouayheb, and D.˜Karpuk, "Gasp codes for secure distributed matrix multiplication," in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 1107–1111.

[65] Z.˜Jia and S.˜A. Jafar, "On the Capacity of Secure Distributed Matrix Multiplication," *arXiv e-prints*, Aug. 2019.

[66] K.˜Banawan and S.˜Ulukus, "Multi-message private information retrieval: Capacity results and near-optimal schemes," *IEEE Transactions on Information Theory*, vol.˜64, no.˜10, pp. 6842–6862, Oct 2018.

[67] R.˜Tajeddine, O.˜W. Gnilke, D.˜Karpuk, R.˜Freij-Hollanti, and C.˜Hollanti, "Private information retrieval from coded storage systems with colluding, byzantine, and unresponsive servers," *IEEE Transactions on Information Theory*, vol.˜65, no.˜6, pp. 3898–3906, June 2019.

[68] M.˜Aliasgari, O.˜Simeone, and J.˜Kliewer, "Distributed and private coded matrix computation with flexible communication load," in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 1092–1096.

[69] S.˜Dutta, M.˜Fahim, F.˜Haddadpour, H.˜Jeong, V.˜Cadambe, and P.˜Grover, "On the optimal recovery threshold of coded matrix multiplication," *arXiv e-prints*, Jan. 2018.

[70] H.˜A. Nodehi and M.˜A. Maddah-Ali, "Limited-sharing multi-party computation for massive matrix operations," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1231–1235.

[71] M.˜Ben-Or, S.˜Goldwasser, and A.˜Wigderson, *Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation.* New York, NY, USA: Association for Computing Machinery, 2019, p. 351–371. [Online]. Available: https://doi.org/10.1145/3335741.3335756

[72] S.˜Lin, T.˜Y. Al-Naffouri, Y.˜S. Han, and W.˜Chung, "Novel polynomial basis with fast fourier transform and its application to reed–solomon erasure codes," *IEEE Transactions on Information Theory*, vol.˜62, no.˜11, pp. 6284–6299, 2016.

[73] Kai Wan, D.˜Tuninetti, and P.˜Piantanida, "On the optimality of uncoded cache placement," in *2016 IEEE Information Theory Workshop (ITW)*, 2016, pp. 161–165.

[74] Z.˜Bar-Yossef, Y.˜Birk, T.˜S. Jayram, and T.˜Kol, "Index coding with side information," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 2006, pp. 197–206.

[75] F.˜Arbabjolfaei, B.˜Bandemer, Y.˜Kim, E.˜Şaşoğlu, and L.˜Wang, "On the capacity region for index coding," in *2013 IEEE International Symposium on Information Theory*, 2013, pp. 962–966.

[76] P.˜Sadeghi, F.˜Arbabjolfaei, and Y.˜Kim, "Distributed index coding," in *2016 IEEE Information Theory Workshop (ITW)*, 2016, pp. 330–334.

[77] M.˜Li, L.˜Ong, and S.˜J. Johnson, "Cooperative multi-sender index coding," *IEEE Transactions on Information Theory*, vol.˜65, no.˜3, pp. 1725–1739, 2019.

[78] O.˜Olmez and A.˜Ramamoorthy, "Constructions of fractional repetition codes from combinatorial designs," in *2013 Asilomar Conference on Signals, Systems and Computers*, 2013, pp. 647–651.

[79] ——, "Fractional repetition codes with flexible repair from combinatorial designs," *IEEE Transactions on Information Theory*, vol.˜62, no.˜4, pp. 1565–1591, 2016.

[80] Q.˜Yan, M.˜Cheng, X.˜Tang, and Q.˜Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Transactions on Information Theory*, vol.˜63, no.˜9, pp. 5821–5833, 2017.

[81] Q.˜Yan, M.˜Wigger, and S.˜Yang, "Placement delivery array design for combination networks with edge caching," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1555–1559.

[82] J.˜Michel and Q.˜Wang, "Placement delivery arrays from combinations of strong edge colorings," *IEEE Transactions on Communications*, vol.˜68, no.˜10, pp. 5953–5964, 2020.

[83] W.˜Song, K.˜Cai, and L.˜Shi, "Coded caching with polynomial subpacketization," *ArXiv*, vol. abs/2001.07020, 2020.

[84] K.˜Shanmugam, A.˜M. Tulino, and A.˜G. Dimakis, "Coded caching with linear subpacketization is possible using ruzsa-szeméredi graphs," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 1237–1241.

[85] C.˜Shangguan, Y.˜Zhang, and G.˜Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Transactions on Information Theory*, vol.˜64, no.˜8, pp. 5755–5766, 2018.

[86] N.˜Mital, D.˜Gündüz, and C.˜Ling, "Coded caching in a multi-server system with random topology," *CoRR*, vol. abs/1712.00649, 2017.

[87] N.˜Mital, K.˜Kralevska, C.˜Ling, and D.˜Gündüz, "Practical functional regenerating codes for broadcast repair of multiple nodes," in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 221–225.

[88] R.˜W. Yeung, "A new outlook on shannon's information measures," *IEEE Transactions on Information Theory*, vol.˜37, no.˜3, pp. 466–474, May 1991.

[89] D.˜Goss, *Basic Structures of Function Field Arithmetic*, ser. Ergebnisse der Mathematik und ihrer Grenzgebeite. 3. Folge. Springer Berlin Heidelberg, 2012. [Online]. Available: https://books.google.co.uk/books?id=KTVqCQAAQBAJ

[90] X.˜Caruso and J.˜Le˜Borgne, "Fast multiplication for skew polynomials," in *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ser. ISSAC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 77–84. [Online]. Available: https://doi.org/10.1145/3087604.3087617

[91] V.˜Strassen, "Gaussian elimination is not optimal," *Numer. Math.*, vol.˜13, no.˜4, p. 354–356, Aug. 1969. [Online]. Available: https://doi.org/10.1007/BF02165411

[92] N.˜J. Bouman and N.˜de˜Vreede, "New protocols for secure linear algebra: Pivoting-free elimination and fast block-recursive matrix decomposition," *IACR Cryptology ePrint Archive*, vol. 2018, p. 703, 2018.

[93] R.˜Cramer and I.˜Damgård, "Secure distributed linear algebra in a constant number of rounds," in *Advances in Cryptology — CRYPTO 2001*, J.˜Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 119–136.

[94] B.˜Hasircioglu, J.˜Gomez-Vilardebo, and D.˜Gunduz, "Bivariate polynomial coding for exploiting stragglers in heterogeneous coded computing systems," 2020.

[95] J.˜Kakar, A.˜Khristoforov, S.˜Ebadifar, and A.˜Sezgin, "Uplink-downlink tradeoff in secure distributed matrix multiplication," *ArXiv*, vol. abs/1910.13849, 2019.

[96] J.˜Kakar, S.˜Ebadifar, and A.˜Sezgin, "Rate-efficiency and straggler-robustness through partition in distributed two-sided secure matrix computation," *ArXiv*, vol. abs/1810.13006, 2018.

[97] A.˜Borodin and R.˜Moenck, "Fast modular transforms," *Journal of Computer and System Sciences*, vol.˜8, no.˜3, pp. 366 – 386, 1974. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000074800292

[98] M.˜Ylinen, A.˜Burian, and J.˜Takala, "Direct versus iterative methods for fixed-point implementation of matrix inversion," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, vol.˜3, 2004, pp. III–225.

[99] "Microsoft SEAL (release 3.5)," https://github.com/Microsoft/SEAL, Apr. 2020, microsoft Research, Redmond, WA.

[100] T.˜Ryffel, A.˜Trask, M.˜Dahl, B.˜Wagner, J.˜Mancuso, D.˜Rueckert, and J.˜Passerat-Palmbach, "A generic framework for privacy preserving deep learning," 2018.