ABSTRACT

CODING AGAINST STRAGGLERS IN DISTRIBUTED COMPUTATION SCENARIOS

by
Malihe Aliasgari

Data and analytics capabilities have made a leap forward in recent years. The volume of available data has grown exponentially. The huge amount of data needs to be transferred and stored with extremely high reliability. The concept of “coded computing”, or a distributed computing paradigm that utilizes coding theory to smartly inject and leverage data/computation redundancy into distributed computing systems, mitigates the fundamental performance bottlenecks for running large-scale data analytics.

In this dissertation, a distributed computing framework, first for input files distributedly stored on the uplink of a cloud radio access network architecture, is studied. It focuses on that decoding at the cloud takes place via network function virtualization on commercial off-the-shelf servers. In order to mitigate the impact of straggling decoders in this platform, a novel coding strategy is proposed, whereby the cloud re-encodes the received frames via a linear code before distributing them to the decoding processors. Transmission of a single frame is considered first, and upper bounds on the resulting frame unavailability probability as a function of the decoding latency are derived by assuming a binary symmetric channel for uplink communications. Then, the analysis is extended to account for random frame arrival times. In this case, the trade-off between an average decoding latency and the frame error rate is studied for two different queuing policies, whereby the servers carry out per-frame decoding or continuous decoding, respectively. Numerical examples demonstrate that the bounds are useful tools for code design and that coding is

instrumental in obtaining a desirable compromise between decoding latency and reliability.

In the second part of this dissertation large matrix multiplications are considered which are central to large-scale machine learning applications. These operations are often carried out on a distributed computing platform with a master server and multiple workers in the cloud operating in parallel. For such distributed platforms, it has been recently shown that coding over the input data matrices can reduce the computational delay, yielding a trade-off between recovery threshold, i.e., the number of workers required to recover the matrix product, and communication load, and the total amount of data to be downloaded from the workers. In addition to exact recovery requirements, security and privacy constraints on the data matrices are imposed, and the recovery threshold as a function of the communication load is studied. First, it is assumed that both matrices contain private information and that workers can collude to eavesdrop on the content of these data matrices. For this problem, a novel class of secure codes is introduced, referred to as secure generalized PolyDot codes, that generalize state-of-the-art non-secure codes for matrix multiplication. Secure generalized PolyDot codes allow a flexible trade-off between recovery threshold and communication load for a fixed maximum number of colluding workers while providing perfect secrecy for the two data matrices. Then, a connection between secure matrix multiplication and private information retrieval is studied. It is assumed that one of the data matrices is taken from a public set known to all the workers. In this setup, the identity of the matrix of interest should be kept private from the workers. For this model, a variant of generalized PolyDot codes is presented that can guarantee both secrecy of one matrix and privacy for the identity of the other matrix for the case of no colluding servers.

CODING AGAINST STRAGGLERS IN DISTRIBUTED COMPUTATION SCENARIOS

by
Malihe Aliasgari

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Helen and John C. Hartmann Department of
Electrical and Computer Engineering

May 2020

APPROVAL PAGE

CODING AGAINST STRAGGLERS IN DISTRIBUTED COMPUTATION SCENARIOS

Malihe Aliasgari

Dr. Jörg Kliewer, Dissertation Advisor Date
Professor, Department of Electrical and Computer Engineering, NJIT

Dr. Osvaldo Simeone, Committee Member Date
Professor, Department of Engineering, Kings College London, U.K.

Dr. Emina Soljanin, Committee Member Date
Professor, Department of Electrical and Computer Engineering, Rutgers - State University of New Jersey, New Brunswick

Dr. Alexander Haimovich, Committee Member Date
Distinguished Professor, Department of Electrical and Computer Engineering, NJIT

Dr. Ali Abdi, Committee Member Date
Professor, Department of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author:	Malihe Aliasgari
Degree:	Doctor of Philosophy
Date:	May 2020

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2020
- Doctor of Philosophy in Pure Mathematics,
  Amirkabir University of Technology, Tehran, Iran, 2014
- Master of Science in Pure Mathematics,
  Amirkabir University of Technology, Tehran, Iran, 2008
- Bachelore of Science in Pure Mathematics,
  Amirkabir University of Technology, Tehran, Iran, 2006

Major: Electrical Engineering

Presentations and Publications:

M. Aliasgari, O. Simeone, and J. Kliewer, “Private and Secure Distributed Matrix Multiplication with Flexible Communication Load,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp 2722-2734, February 2020.

M. Aliasgari, O. Simeone, and J. Kliewer, “Distributed and Private Coded Matrix Computation with Flexible Communication Load,” *IEEE International Symposium on Information Theory (ISIT)*, pp 1092-1096, 2019.

M. Aliasgari, J. Kliewer,and O. Simeone, “Coded Computation Against Processing Delays for Virtualized Cloud-Based Channel Decoding,” *IEEE Transactions on Communications*, vol. 67, pp 28-38, January 2019.

M. Aliasgari, J. Kliewer, and O. Simeone, “Coded computation against straggling decoders for network function virtualization,” *IEEE International Symposium on Information Theory (ISIT)*, pp 711-715, June 2018.

تقدیم با عشق به

پدر بزرگوار و مادر مهربانم، همسر عزیز و همیشه همراهم و فرزند دلبند و نازنینم

زندگی صحنه ی یکتای هنرمندی ماست
هر کسی نغمه ی خود خواند و از صحنه رود
صحنه پیوسته به جاست
خرم آن نغمه که مردم بسپارند به یاد

ژاله اصفهانی

DEDICATED TO MY BELOVED PARENTS,
MY LOVELY HUSBAND,
AND MY DEAREST SON

Life is a stage, we play our own unique roles
Everyone sings his own melody, leaves scene then after
Life is an everlasting stage
Great those melodies that people recall

Jaleh Esfahani

ACKNOWLEDGMENT

I would like to thank my advisor, Professor Jörg Kliewer for his advising and financial support.

My dissertation committee member Professor Osvaldo Simeone has an invaluable part of my experience. The seeds of this dissertation were planted during the December 2016, after passing my qualify exam, when we had a meeting with Professor Simeone. I was lucky to be advised by Professor Simeone, from whom I learned from his insight, vision, commitment, patient, and forgiveness. He asked sharp and insightful questions during our meetings that helped shape several key results presented in this dissertation. I cannot thank him enough for his help and advising.

I would like to thank Professor Emina Soljanin, Professor Alexander Haimovich, and Professor Ali Abdi for serving on my dissertation committee, and Professor Haim Grebel and Nirwan Ansari for serving on my qualifying examination committee.

I would also like to thank all the faculty members who I interacted with over these years, in particular, Professor Walid Hubbi, Professor Reza Khanbilvardi, Professor Daniel Panario, Professor Pulkit Grover, Professor Edgar Martinez-Moro Professor Karl-Heinz Zimmermann, and Professor Daniel Cabarcas.

Above all, I would like to sincerely thank my precious family. I am eternally grateful to my mother, Zahra Safaei, and my father, Mahmoudreza Aliasgari, who have always been there for me and supported me to overcome the challenges and obstacles. I am very thankful to my kind brothers, Mojib and Moein, who have always supported me emotionally and wanted the best for me.

My deep and sincere gratitude towards my better half Dr. Yousef Nejatbakhsh for his eternal support, unfailing and continuous love for encouraging me in all of my pursuits, and inspiring me to follow my dreams. He always stay with me, listen to me, and help me in times of despair, sorrow, and boredom and in times of hope, joy and triumph. His patience and sacrifice will remain my inspiration throughout my life. Without his help, I would not have been able to become who I am.

Last, but not least, thank you God for new life of my baby, the most precious jewels from you, which changes my life and hope for the best, my dearest son, OMID.

TABLE OF CONTENTS

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES (Continued)

CHAPTER 1

INTRODUCTION

For over fifty years, people have been trying to follow Moores law and to increase the number of transistors on a circuit by making the transistors smaller. However, this trend is going end sometime inevitably, due to the limit imposed by power density issues. There may be other types of breakthrough technologies that can make the computation components even smaller and faster, but the main point is that the fundamental theory of transistors may not apply any more to the new devices and platforms. The inevitable saturation of Moores law has made researchers look into the possibility and increasing size and dimension of data, system designers have increasingly resorted to parallel and distributed computing to reduce the computation time of machine-learning algorithms.

Distributed computing has become the basis for all large-scale computation. The ability to distribute work over many processors and utilize their combined computational power to run large tasks in parallel has become necessary with the increasing size of data sets and task complexity.

All these breakthroughs were made possible only due to the scalability in computing and storage capacity offered by modern large-scale clusters. While classical computing and storage clusters were composed of a small number of expensive, custom-designed high-end machines, modern large-scale clusters consist of more than tens of thousands of hardware nodes, connected through general-purpose network infrastructure.

In order to develop and deploy sophisticated solutions and tackle large-scale problems in machine learning, science, and engineering, it is important to understand

and optimize novel and complex trade-offs across the multiple dimensions of computation, communication, and storage.

One of the most serious issues facing distributed computing is that processors take randomly varying amounts of time to finish, which means that all too frequently, when a task is divided into N parts amongst N processors, several will straggle. In these traditional schemes, one needs to wait for all processors to finish computing before the job is done, resulting in increasing latency. Most current research in this domain focuses on replicating the work of straggling processors by giving their job to processors that have already finished. However, these approaches increase communication and coordination cost, two bottlenecks in most systems. We tackle the straggler problem arising in distributed computing by assigning redundant computations to nodes derived through error correcting codes.

In this dissertation, distributed computing systems are viewed through a coding-theoretic lens. The role of codes in providing resiliency against noise has been studied for decades in many other engineering contexts, especially in communication systems, and is part of our everyday infrastructure (smartphones, laptops, WiFi and cellular systems, etc.). Since the performance of distributed systems is also significantly affected by the system behavior and bottlenecks, which we call “system noise”, there is an exciting opportunity for codes to endow distributed systems with robustness against such system noise.

1.1 Main Contribution of this Dissertation

The backbone of this dissertation consists of two main parts: coded computation for network function virtualization and coded computation for large matrix multiplication. The role of codes for each of these parts is addressed in this dissertation, and the main contributions are summarized as follows.

- The first contribution of the dissertation is **to propose coded computing to enable reliable and timely channel decoding in a cloud radio access network architecture based on distributed unreliable processors.** Codes transform the evolution of large-scale distributed storage systems in modern data centers, which have a major impact on industry. The implementation of channel decoding in the cloud by means of network function virtualization is faced with the challenge of providing reliable operation despite the unreliability of commercial off-the-shelf servers. The proposed coded network function virtualization solution leverages the algebraic structure of the transmitted coded data frames in order to enhance the robustness of channel decoding constraint.

- Runtime performance of distributed algorithms is heavily affected by stragglers, i.e., "processors lagging behind in the execution of a certain orchestrated function", which **we show to significantly improve the latency by using coded computation against processing delays for network function virtualization.**

- Linear codes are used in forward error correction and are applied in methods for transmitting symbols (e.g., bits) on a communications channel so that, if errors occur in the communication, some errors can be corrected or detected by the recipient of a message block. As a byproduct of the analysis **we introduce the dependency graph of a linear code and its chromatic number as novel relevant parameters of a linear code** beside minimum distance, blocklength, and rate.

- Fast content download is one of the major user demands. Content download time includes the time taken for a user to compete with the other users to access to the processors, and the time to acquire the data from the processors. In this dissertation a transmission of a single frame is considered first. Then, **we analyze random frame arrival times in network function virtualization.** In this case, the trade-off between an average decoding latency and the frame error rate is studied for two different queuing policies, whereby the servers carry out per-frame decoding or continuous decoding.

- A rapid growth of large-scale machine learning and big data analytics, facilitating the developments of data-intensive applications. Faced with the saturation of Moores law and increasing size and dimension of data, system designers have increasingly resorted to parallel and distributed computing to reduce computation time of machine-learning algorithms. For computing large matrix multiplications, **we introduce a novel perspective on distributed computing codes based on the signal processing concepts of convolution and z-transform.**

- Finally, the phenomenal growth in computing power over much of the past five decades has been motivated by scientific applications demanding, high-performance parallel computing. The data tends to be distributed, and issues such as privacy and security. To this end, **we impose security and privacy constraints on large data matrices, and study the recovery threshold as a function of the communication load.**

1.2 Related Works

In this section, we provide a high-level overview of some selected related works. More extensive surveys of the related works are provided in the following chapters.

1.2.1 Distributed Coded Computation

The era of Big Data and the immensity of real-life datasets compels computation tasks to be performed in a distributed fashion, where the data is dispersed among many servers that operate in parallel. Coding has always played a fundamental role in information propagation, e.g., in communication systems [95] and storage systems [30]. In this dissertation, we aim to develop this understanding of coding theory and techniques from communication systems to the much broader field of computation system.

Sometimes, the code is applied in such a way that the communicated messages, instead of the data, are encoded, for instance in, [66, 71, 102, 111, 36, 13, 72, 7, 6, 8, 84]. In this case, the data is essentially replicated, and the replication factor, which relates to the density of the code, directly determines the overhead in both storage and computation time. Although the main observation is that the communication time may dominate the overall time cost, and hence increasing the computation time may be acceptable, it is desirable that the overall storage cost does not increase by much. Thus, a sparse code is more desirable than a dense code.

1.2.2 Straggler Mitigation

Stragglers are one of the main reasons why the actual speed-up of a parallelized computation is always worse than the theoretical speed-up predicted by Amdahl's

Law [10, 86]. A straggler is a processor which is still working on its share of the parallelized computation when most or all of the other processors have completed their shares. It is shown that running a computational task at parallel servers involves unpredictable latency due to several factors such as network latency, shared resources, maintenance activities, and power limits [12, 28]. Moreover, the stragglers cannot be completely removed from a distributed computing cluster. One approach to mitigate the adverse effect of stragglers is based on efficient straggler detection algorithms. For instance, in order to combat with stragglers, cloud computing frame works like Hadoop [96] employ various straggler detection techniques and usually reset the task allotted to stragglers. Another line of approaches is based on forward error-correction techniques offer an alternative approach to deal with this straggler effect by introducing redundancy in the computational tasks across different processors. The fusion or master node now requires outputs from only a subset of all the processors to successfully finish. The use of preliminary erasure codes dates back to the ideas of algorithmic fault tolerance [33]. Adding redundancy has also been proposed as a way to tackle the straggler problem: by replicating tasks the runtime of distributed algorithms can be significantly improved [62, 11, 93, 67, 42, 106, 24]. By collecting outputs of the fast-responding nodes (and potentially cancelling all the other slow-responding replicas), such replication-based scheduling algorithms can reduce latency.

1.2.3 Distributed Matrix Multiplication

At the core of many signal processing, machine learning applications, scientific computing, and graph processing are tensor operations, most notably large matrix multiplications [51]. Many such applications require processing terabytes or even petabytes of data, which needs massive computation and storage resources that cannot be provided by a single machine. Hence, deploying matrix computation tasks on large-scale distributed systems has received wide interest [21, 25, 104, 97]. There is also a lot of interest in classical Algorithm-Based Fault Tolerance literature, e.g., [49, 45] and more recently in distributed coded computation literature, to make matrix multiplications resilient to faults and delays. An important problem in distributed matrix multiplication is to consider the case where the inputs are encoded and multiplied in a block-wise manner. This setup generalizes the problem formulated in [111] to enable a more flexible trade-off between resources such as storage, computation and communication,and has been studied in [68, 112, 39, 8, 81, 52].

1.2.4 Secret Sharing Schemes

Secret sharing refers to any method for distributing a secret among a group of participants, each of which allocates a share of the secret. The secret can only be reconstructed when the shares are combined together; individual shares are of no use on their own. Secret-sharing schemes were introduced by Blakley [18] and Shamir [94] independently for the threshold case, that is, for the case where the subsets that can reconstruct the secret are all the sets whose cardinality is at least a certain threshold.

The volume of data continues to rapidly grow as information pours from various platforms. This has motivated the fast development of scalable, interpretable, and fault-tolerant distributed computing frameworks. In distributed computing systems Over a general network, all communication between the fusion node and a participant or processors, who is not directly connected to it, must pass through other participants in the network. The fact that often the data is confidential and processors are curious and untrusted poses the challenge of secret sharing over a network without leaking any additional information to any processors [81, 113, 22, 56, 107, 27].

1.2.5 Private Information Retrieval

The most efficient way for a user to retrieve a desired message from a set of distributed servers, each of which stores all the messages, without revealing any information about which message is being retrieved to any individual server is given as the private information retrieval problem. The user can hide his interests trivially by requesting all the information, but that could be very inefficient (expensive). The goal of the private information retrieval problem is to find the most efficient solution in terms of download complexity. The private information retrieval problem was introduced in 1995 [25, 26] and of broad interest because it shares intimate connections to many other prominent problems [44, 40, 3, 15, 16, 94]. Private information retrieval also connects distributed data storage repair [31], index coding [17] and matrix multiplication [108]. As such, private information retrieval holds tremendous promise as a point of convergence of complementary perspectives.

1.3 Dissertation Outline

The rest of the thesis is organized as follows:

Chapter 2. We consider the uplink of a cloud radio access network architecture in which decoding at the cloud takes place via network function virtualization on commercial off-the-shelf servers. In order to mitigate the impact of straggling decoders in this platform, a novel coding strategy is proposed, whereby the cloud re-encodes the received frames via a linear code before distributing them to the decoding processors. Transmission of a single frame is considered first, and upper bounds on the resulting frame unavailability probability as a function of the decoding latency are derived by assuming a binary symmetric channel for uplink communications. Then, the analysis is extended to account for random frame arrival times. In this case, the trade-off between an average decoding latency and the frame error rate is studied for two different queuing policies, whereby the servers carry out per-frame decoding or continuous decoding, respectively. Numerical examples demonstrate that the bounds are useful tools for code design and that coding is instrumental in obtaining a desirable compromise between decoding latency and reliability.

Chapter 3. In addition to the exact recovery requirements of two large matrix multiplications, we impose security and privacy constraints on the data matrices, and study the recovery threshold as function of the communication load. We first assume that both matrices contain private information and that workers can collude to eavesdrop on the content of these data matrices. We introduce a novel class of secure codes, referred to as secure generalized PolyDot codes, that generalize state-of-the-art non-secure codes for matrix multiplication. We then study a connection between secure matrix multiplication and private information retrieval. For this model, we present a variant of generalized PolyDot codes that can guarantee both secrecy of one matrix and privacy for the identity of the other matrix for the case of no colluding servers.

Chapter 4. We conclude the dissertation with the summary of the results and important future research directions.

CHAPTER 2

CODED COMPUTATION AGAINST PROCESSING DELAYS IN NETWORK FUNCTION VIRTUALIZATION

2.1 Introduction

Promoted by the European Telecommunications Standards Institute (ETSI), network function virtualization has become a cornerstone of the envisaged architecture for 5G systems [77]. Network function virtualization leverages virtualization technologies in order to implement network functionalities on commercial off-the-shelf programmable hardware, such as general purpose servers, potentially reducing both capital and operating costs. An important challenge in the deployment of network function virtualization is ensuring carrier grade performance while relying on commercial off-the-shelf components. Such components may be subject to temporary unavailability due to malfunctioning, and are generally characterized by randomness in their execution runtimes. The typical solution to these problems involves replicating the virtual machines that execute given network functions on multiple processors, e.g., cores or servers [1, 75, 46, 57].

Among the key applications of network function virtualization is the implementation of centralized radio access functionalities in a cloud radio access network [78, 2]. As shown in Figure 2.1, each remote radio head of a cloud radio access network architecture is connected to a cloud processor by means of a fronthaul link. Baseband functionalities are carried out on a distributed computing platform in the cloud, which can be conveniently programmed and reconfigured using network function virtualization. The most expensive baseband function in terms of latency to be carried out at the cloud is uplink channel decoding [78, 9, 79].

The implementation of channel decoding in the cloud by means of network function virtualization is faced not only with the challenge of providing reliable operation despite the unreliability of commercial off-the-shelf servers, but also with the latency constraints imposed by retransmission protocols. In particular, keeping decoding latency at a minimum is a major challenge in the implementation of cloud radio access network owing to timing constraints from the link-layer retransmission protocols [34, 90, 60]. In fact, positive or negative feedback signals need to be sent to the users within a strict deadline in order to ensure the proper operation of the protocol. In [87, 88] it is argued that exploiting parallelism across multiple cores in the cloud can reduce the decoding latency by enabling decoding as soon as one can has computed its task. However, parallel processing does not address the unreliability of commercial off-the-shelf hardware. A different solution is needed in order to address both unreliability and delays associated with cloud decoding.

The problem of straggling processors, that is, of processors lagging behind in the execution of a certain orchestrated function, has been well studied in the context of distributed computing [29, 12, 114, 72, 71, 70]. Recently, it has been pointed out that, for the important case of linear functions, it is possible to improve over repetition strategies in terms of the trade-off between performance and latency by carrying out linear precoding of the data prior to processing, e.g., [65, 73, 109, 101, 36, 92, 111, 76, 63]. The key idea is that, by employing suitable linear (erasure) block codes operating over fractions of size $1/K$ of the original data, a function may be completed as soon as any K or more processors, depending on the minimum distance of the code, have completed their operations. Coding has also been found to be useful addressing the straggler problem in the context of coded distributed storage and computing systems, see, e.g., [106, 55, 11, 108, 4].

In this dissertation, we explore the use of coded computing to enable reliable and timely channel decoding in a cloud radio access network architecture based on distributed unreliable processors. Specifically, we formally and systematically address the analysis of coded network function virtualization for cloud radio access network uplink decoding. The only prior work on coded computing for network function virtualization is [5], which provides numerical results concerning a toy example with three processors in which a processor in the cloud is either on or off. Unlike [5], in this work, we derive analytical performance bounds for a general scenario with any number of servers, random computing runtimes, and random packet arrivals. Specific novel contributions are as follows.

- We first consider the transmission of an isolated frame, and develop analytical upper bounds on the frame unavailability probability as a function of the allowed decoding delay. The frame unavailability probability measures the probability that a frame is correctly decoded within a tolerated delay constraint. The frame unavailability probability bounds leverage large deviation results for correlated variables [50] and depend on the properties of both the uplink linear channel code adopted at the user and the network function virtualization linear code applied at the cloud;
- As a byproduct of the analysis we introduce the dependency graph of a linear code and its chromatic number as novel relevant parameters of a linear code beside minimum distance, blocklength, and rate;
- We extend the analysis to account for random frame arrival times, and investigate the trade-off between average decoding latency and frame error rate (FER) for two different queuing policies, whereby the servers carry out either per-frame or continuous decoding;
- We provide extensive numerical results that demonstrate the usefulness of the derived analytical bounds in both predicting the system performance and enabling the design of network function virtualization codes.

The rest of this chapter is organized as follows. In Section 2.2, we review some technical background and preliminaries for the rest of this chapter. In Section 3.2, we present the system model focusing, as in [5], on a binary symmetric channel (BSC) for uplink communications. Section 2.4 presents the two proposed upper bounds on the frame unavailability probability as a function of latency. In Section 2.5 we study the proposed system with random frame arrival times, and Section 2.6 provides numerical results.

2.2 Technical Background and Preliminaries

2.2.1 Large Deviation for Dependent Random Variables

Many random variables can be written as a sum

$$X = \sum_{\alpha \in \mathcal{A}} Y_\alpha \tag{2.1}$$

of simpler random variables Y_α, with ranging over some index set. For example, each Y_α may be an indicator variable taking the values 0 and 1 only, i.e. $Y_\alpha \sim \text{Bern}(p_\alpha)$ for some $p_\alpha \in [0, 1]$. In this dissertation, we are interested in situations where the variables Y_α may be dependent, but there is a large amount of independence among them. A typical situation is the sum

$$X = \sum_{i_1 \cdots i_d} f_{i_1 \cdots i_d}(\xi_{i_1}, \ldots, \xi_{i_d}), \tag{2.2}$$

for some functions $f_{i_1 \cdots i_d}$ and independent random variables $\xi_{i_1}, \ldots, \xi_{i_d}$, and some set $\mathcal{A} \subseteq [n]^d_<$, where $[n]^d_<$ is the set of all d-tuples $(i_1, \ldots, i_d)$ with $i_1 < \cdots < i_d \leq n$. Here d and n are some positive integers; typically d is small (perhaps only 2 or 3) and n is large. One example of such sums (2.2) is the family of U-statistics [47], which is the symmetric case obtained by taking $\xi_{i_1}, \ldots, \xi_{i_d}$ independent and identically distributed, all $f_{i_1 \cdots i_d}$ equal to some symmetric function f, and $\mathcal{A} = [n]^d_<$. More generally, if we in this situation sum over a subset $\mathcal{A} \subset [n]^d_<$, we obtain an incomplete U-statistic. Also two sample U-statistics are of the general type (2.2), but now the ξ_i are of two different types.

Definition 1. *A dependency graph is a directed graph representing dependencies of several objects towards each other. It is possible to derive an evaluation order or the absence of an evaluation order that respects the given dependencies from the dependency graph. More preciecly, for random variables Y_α, $\alpha \in \mathcal{A}$, a graph Γ with vertex set $\mathcal{A}$ such that if $\mathcal{B} \subset \mathcal{A}$ and $\alpha \in \mathcal{A}$ is not connected by an edge to any vertex in $\mathcal{B}$, then Y_α is independent of $\{Y_\beta\}_{\beta \in \mathcal{B}}$.*

Definition 2. *The chromatic number $\chi(\Gamma)$ of a graph Γ is the smallest number of colors needed to color the vertices of Γ so that no two adjacent vertices share the same color [105, page 334].*

We can now state some results of [50], which we use in this chapter.

Theorem 1. *Suppose that X is as in (2.1) with a $a_\alpha \leq Y_\alpha \leq b_\alpha$ for every $\alpha \in \mathcal{A}$ and some real numbers a_α and b_α. Then for $t > 0$,*

$$\mathbb{P}(X \geq \mathbb{E}X + t) \leq \exp\left(-2\frac{t^2}{\chi(\Gamma)\sum_{\alpha\in\mathcal{A}}(b_\alpha - a_\alpha)^2}\right) \tag{2.3}$$

The same estimate holds for $\mathbb{P}(X \geq \mathbb{E}X - t)$.

Corollary 1. *Suppose that X is as in (2.1) with $Y\alpha \sim Be(p_\alpha)$ for some $p_\alpha \in (0,1)$ and all $\alpha \in \mathcal{A}$. Then, for $t \geq 0$,*

$$\mathbb{P}(X \geq \mathbb{E}X + t) \leq \exp\left(-2\frac{t^2}{\chi(\Gamma)|\mathcal{A}|}\right). \tag{2.4}$$

The same estimate holds for $\mathbb{P}(X \geq \mathbb{E}X - t)$.

These results can be improved by Hoeffding's methods [47, 48], when the summands have variances that are substantially smaller than the upper bound $(b_\alpha - a_\alpha)^2/4$. The following results holds for $\mathbb{P}(X \leq \mathbb{E}X - t)$. If the boundedness assumption is reversed to $Y_\alpha - \mathbb{E}Y_\alpha \geq -b$.

Theorem 2. *Suppose that X is as (2.1) with $Y_\alpha - \mathbb{E}Y_\alpha \leq b$ for some $b > 0$ and all $\alpha \in \mathcal{A}$. Then, with $\varphi(x)$ defined as follows*

$$\varphi(x) \triangleq (1 + x)\ln(1 + x) - x, \tag{2.5}$$

and $S \triangleq \sum_{\alpha\in\mathcal{A}}$ Var Y_α, for $t \geq 0$,

$$\begin{aligned}\mathbb{P}(X \geq \mathbb{E}X + t) &\leq \exp\left(-\frac{S}{b^2\chi(\Gamma)}\varphi\left(\frac{4bt}{5S}\right)\right) &(2.6)\\ &\leq \exp\left(-\frac{8t^2}{25\chi(\Gamma)(S + bt/3)}\right). &(2.7)\end{aligned}$$

Corollary 2. *Suppose that X is as (2.1) with $Y_\alpha \sim Be(p)$ for some $p \in (0,1)$ and all $\alpha \in \mathcal{A}$. Let $N \triangleq |\mathcal{A}|$. Then for $t \geq 0$,*

$$\begin{aligned}
\mathbb{P}(X \geq \mathbb{E}X + t) &\leq \exp\left(-\frac{Np}{(1-p)\chi(\Gamma)}\varphi\left(\frac{4t}{5Np}\right)\right) && (2.8)\\
&\leq \exp\left(-\frac{8t^2}{25\chi(\Gamma)(Np + t/3)}\right); && (2.9)\\
\mathbb{P}(X \leq \mathbb{E}X - t) &\leq \exp\left(-\frac{N(1-p)}{p\chi(\Gamma)}\varphi\left(\frac{4t}{5N(1-p)}\right)\right) && (2.10)\\
\mathbb{P}(X \leq \mathbb{E}X - t) &\leq \exp\left(-\frac{8t^2}{25\chi(\Gamma)Np}\right). && (2.11)
\end{aligned}$$

If a dependency graph Γ is given, then we may replace $\chi(\Gamma)$ by $\Delta(\Gamma)$, where $\Delta(\Gamma)$ denote the maximum degree of Γ. It can be easily seen that $\chi(\Gamma) \leq \Delta(\Gamma) + 1$, [19].

2.2.2 Queueing Theory

In this subsection, we review some concepts of a single server exponential queueing theory which we use in Section 2.5.

Suppose that customers arrive at a single-server service station in accordance with a Poisson process having rate λ. That is, the times between successive arrivals are independent exponential random variables having mean $1/\lambda$. Each customer, upon arrival, goes directly into service if the server is free and, if not, the customer joins the queue. When the server finishes serving a customer, the customer leaves the system, and the next customer in line, if there is any, enters service. The successive service times are assumed to be independent exponential random variables having mean $1/\mu$.

The preceding is called the $M/M/1$ queue. The two Ms refer to the fact that both the inter arrival and the service distributions are exponential (and thus, memoryless, or Markovian), and the 1 to the fact that there is a single server.

Definition 3. *The $M/G/1$ model assumes Poisson arrivals at rate λ; a general service distribution; and a single server. In addition, we will suppose that customers are served in the order of their arrival.*

In parallel computing, the forkjoin model is a way of setting up and executing parallel programs, such that execution branches off in parallel at designated points in the program, to join (merge) at a subsequent point and resume sequential execution. Parallel sections may fork recursively until a certain task granularity is reached. Forkjoin can be considered a parallel design pattern [82]. This model is also called as (n, n) fork-joint system.

Definition 4. *[54] An (n, k) fork-join system consists of n nodes. Every arriving job is divided into n tasks which enter first-come first-serve queues at each of the n nodes. The job departs the system when any k out of n tasks are served by their respective nodes. The remaining nk tasks abandon their queues and exit the system before completion of service.*

Definition 5. *(Expected Latency). The expected latency $\mathbb{E}[T]$ is defined as the expected time from the arrival of a task until any one replica is served. It includes the waiting time in queue and the time spent at the servers until the task is served.*

Although $\mathbb{E}[T]$ is a good indicator of the average behavior, system designers are often interested in the tail $Pr(T > t)$ of the latency. For many queueing problems, determining the distribution of response time T requires the assumption of exponential service time.

2.3 System Model

As illustrated in Figure 2.1, we consider the uplink of a cloud radio access network system in which a user communicates with the cloud via a remote radio head (RRH). The user is connected to the remote radio head via a BSC with cross error probability

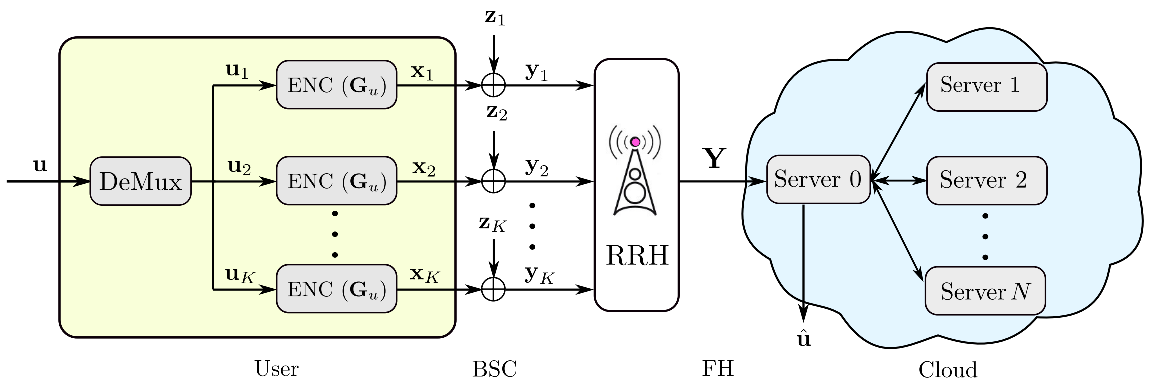


Figure 2.1 Network function virtualization model for uplink channel decoding. The input information frame $\mathbf{u}$ is divided into packets, which are encoded with a linear code $\mathcal{C}_u$ with generator matrix $\mathbf{G}_u$. The packets are received by the remote radio head (RRH) through a BSC and forwarded to the cloud. Server 0 in the cloud re-encodes the received packet with a linear code $\mathcal{C}_c$ in order to enhance the robustness against potentially straggling Servers $1, \ldots, N$.

δ, while the remote radio head-to-cloud link, typically referred to as fronthaul, is assumed to be noiseless. Note that the BSC is a simple model for the uplink channel, while the noiseless fronthaul accounts for a typical deployment with higher capacity fiber optic cables. The analysis can be generalized to other additive noise channel, such as Gaussian channels. The cloud contains a master server, or Server 0, and N slave servers, i.e., Servers $1, \ldots, N$. The slave servers are characterized by random computing delays as in related works on coded computation [65, 73, 92]. Note that we use here the term "server" to refer to a decoding processor, although, in a practical implementation, this may correspond to a core of the cloud computing platform [87, 88].

In the first part of this chapter, we consider transmission of a single information frame $\mathbf{u}$, while Section 2.5 focuses on random frame arrival times and queuing effect delays. The user encodes an information frame $\mathbf{u}$ consisting of L bits. Before encoding, the information frame is divided into K blocks $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K \in \{0,1\}^{L/K}$ of equal size, each of them containing L/K bits. As shown in Figure 2.1, in order to combat noise on the BSC, the L/K blocks are encoded by an (n,k) binary linear

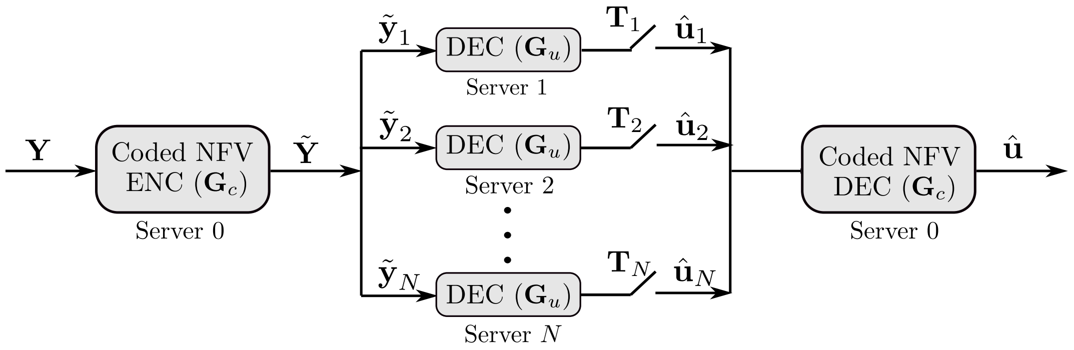


Figure 2.2 Coded Network function virtualization at the cloud: Server 0 re-encodes the received packets in $\mathbf{Y}$ by a linear network function virtualization code $\mathcal{C}_c$ with generator $\mathbf{G}_c$. Each encoded packet $\tilde{\mathbf{y}}_i$ is then conveyed to Server i for decoding.

code $\mathcal{C}_u$ of rate $r = k/n$ defined by generator matrix $\mathbf{G}_u \in \mathbb{F}_2^{n\times k}$, where $n = L/(rK)$ and $k = L/K$. Let $\mathbf{x}_j \in \{0,1\}^n$ with $j \in \{1,\ldots,K\}$ be the K transmitted packets of length n. At the output of the BSC, the length-n received vector for the jth packet at the remote radio head is given as

$$\mathbf{y}_j = \mathbf{x}_j \oplus \mathbf{z}_j, \tag{2.12}$$

where $\mathbf{z}_j$ is a vector of i.i.d. Bern(δ) random variables (rvs). The K received packets $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K)$ by the remote radio head are transmitted to the cloud via the fronthaul link, and the cloud performs decoding. Specifically, as detailed next, we assume that each Server $1,\ldots,N$ performs decoding of a single packet of length n bits while Server 0 acts as coordinator.

Assuming $N \geq K$, we adopt the idea of network function virtualization coding proposed in [5]. Accordingly, as seen in Figure 2.2, the K packets are first linearly encoded by Server 0 into $N \geq K$ coded blocks of the same length n bits, each forwarded to a different server for decoding. This form of encoding is meant to mitigate the effect of straggling servers in a manner similar to [65, 73, 92]. Using an (N,K) binary linear network function virtualization code $\mathcal{C}_c$ with $K \times N$ generator matrix $\mathbf{G}_c \in \mathbb{F}_2^{N\times K}$, the encoded packets are obtained as

$$\tilde{\mathbf{Y}} = \mathbf{Y}\mathbf{G}_c, \tag{2.13}$$

where $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_K]$ is the $n \times K$ matrix obtained by including the received signal $\mathbf{y}_j$ as the jth column and $\tilde{\mathbf{Y}} = [\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_N]$ is the $n \times N$ matrix whose ith column $\tilde{\mathbf{y}}_i$ is the input to Server i, where $i \in \{1, \ldots, N\}$. From (2.12), this vector can be written as

$$\tilde{\mathbf{y}}_i = \sum_{j=1}^{K} \mathbf{y}_j g_{c,ji} = \sum_{j=1}^{K} \mathbf{x}_j g_{c,ji} + \sum_{j=1}^{K} \mathbf{z}_j g_{c,ji}, \tag{2.14}$$

where $g_{c,ji}$ is the (j,i)th entry of matrix $\mathbf{G}_c$.

The signal part $\sum_{j=1}^{K} \mathbf{x}_j g_{c,ji}$ in (2.14) is a linear combination of d_i codewords for the rate-r binary code with generator matrix $\mathbf{G}_u$, and hence, it is a codeword of the same code. The parameter d_i, $i \in \{1, \ldots, N\}$, denotes the Hamming weight of the ith column of matrix $\mathbf{G}_c$, where $0 \leq d_i \leq K$. Each server i receives as input $\tilde{\mathbf{y}}_i$ from which it can decode the codeword $\sum_{i=1}^{K} \mathbf{x}_i g_{c,ji}$. This decoding operation is affected by the noise vector $\sum_{j=1}^{K} \mathbf{z}_j g_{ji}$ in (2.14), which has i.i.d. Bern(γ_i) elements. Here, γ_i is obtained as the first row and second column's entry of the matrix $\mathbf{Q}^{d_i}$, with $\mathbf{Q}$ being the transition matrix of the BSC with crossover probability δ, i.e.,

$$\mathbf{Q} = \begin{bmatrix} 1-\delta & \delta \\ \delta & 1-\delta \end{bmatrix}. \tag{2.15}$$

As an example, $d_i = 2$, implies a bit flipping probability of $\gamma_i = 2\delta(1-\delta)$. Note that a larger value of d_i yields a larger bit probability γ_i. We define as $\mathrm{P}_{n,k}(\gamma_i)$ the decoding error probability of the (n,k) linear user code at Server i, which can be upper bounded by using [83, Theorem 33].

Server i requires a random time $T_i = T_{1,i} + T_{2,i}$ to complete decoding, which is modeled as the sum of a component $T_{1,i}$ that is independent of the workload and a component $T_{2,i}$ that instead grows with the size n of the packet processed at each server, respectively. The first component accounts, e.g., for processor unavailability periods, while the second models the execution runtime from the start of the computation. The first variable $T_{1,i}$ is assumed to have an exponential probability density function (pdf) $f_1(t)$ with mean $1/\mu_1$, while the variable $T_{2,i}$ has a shifted exponential distribution with cumulative distribution function (cdf) [84]

$$F_2(t) = 1 - \exp\left(-\frac{rK\mu_2}{L}\left(t - a\frac{L}{rK}\right)\right), \tag{2.16}$$

for $t \geq aL/(rK)$ and $F_2(t) = 0$ otherwise. The parameter a represents the minimum processing time per input bit, while $1/\mu_2$ is the average additional time needed to process one bit. As argued in [65, 84], the shifted exponential model provides a good fit for the distribution of computation times over cloud computing environments such as Amazon EC2 clusters. The cdf of the time T_i can hence be written as the integral $F(t) = \int_0^t f_1(\tau)F_2(t-\tau)d\tau$. We also assume that the runtime rvs $\{T_i\}_{i=1}^N$ are mutually independent. Due to (2.16), the probability that a given set of l out of N servers has finished decoding by time t is given as

$$a_l(t) = \binom{N}{l} F(t)^l (1 - F(t))^{N-l}. \tag{2.17}$$

Let $d_{\min}$ be the minimum distance of the network function virtualization code $\mathcal{C}_c$. Due to (2.14), Server 0 in the cloud is able to decode the message $\mathbf{u}$ or equivalently the K packets $\mathbf{u}_j$ for $j \in \{1, \ldots, K\}$, as soon as $N - d_{\min} + 1$ servers have decoded successfully. Let $\hat{\mathbf{u}}_i$ be the output of the ith server in the cloud upon decoding. We assume that an error detection mechanism, such as a cyclic redundancy check (CRC), is in place so that Server 0 outputs

$$\hat{\mathbf{u}}_i = \begin{cases} \hat{\mathbf{u}}_i, & \text{for correct decoding,} \\ \emptyset, & \text{otherwise.} \end{cases}$$

The output $\hat{\mathbf{u}}(t)$ of the decoder at Server 0 at time t is then a function of the vectors $\hat{\mathbf{u}}_i(t)$ for $i \in \{1, \ldots, N\}$, where

$$\hat{\mathbf{u}}_i(t) = \begin{cases} \hat{\mathbf{u}}_i, & \text{if } T_i \leq t, \\ \emptyset, & \text{otherwise.} \end{cases}$$

Finally, the frame unavailability probability at time t is defined as the probability

$$\mathrm{P}_u(t) = \Pr\left[\hat{\mathbf{u}}(t) \neq \mathbf{u}\right]. \tag{2.18}$$

The event $\{\hat{\mathbf{u}}(t) \neq \mathbf{u}\}$ occurs when either not enough servers have completed decoding or many servers have completed but failed decoding by time t. We also define the FER as

$$\mathrm{P}_e = \lim_{t \to \infty} \mathrm{P}_u(t). \tag{2.19}$$

The FER measures the probability that, when all servers have completed decoding, a sufficiently large number, namely larger than $N - d_{\min}$, has decoded successfully.

2.4 Bounds on the Frame Unavailability Probability

In this section, we derive analytical bounds on the frame unavailability probability $\mathrm{P}_u(t)$ in (2.18) as a function of the decoding latency t.

2.4.1 Preliminaries

Each server i with $i \in \{1, \ldots, N\}$ decodes successfully its assigned packet $\tilde{\mathbf{y}}_i$ if: (i) the server completes decoding by time t; (ii) the decoder at the server is able to correct the errors caused by the BSC. Furthermore as discussed, an error at Server 0 occurs at time t if the number of servers that have successfully decoded by time t is smaller than $N - d_{\min} + 1$.

To evaluate the frame unavailability probability, we hence define the indicator variables $C_i(t) = \mathbb{1}\{T_i \leq t\}$ and D_i which are equal to 1 if the events (i) and (ii)

described above occur, respectively, and zero otherwise. Based on these definitions, the frame unavailability probability is equal to

$$\mathrm{P}_u(t) \quad = \quad \Pr\left[\sum_{i=1}^{N} C_i(t)D_i \leq N - d_{\min}\right]. \tag{2.20}$$

The indicator variables $C_i(t)$ are independent Bernoulli rvs across the servers $i \in \{1, \ldots, N\}$, due to the independence assumption on the rvs T_i. However, the indicator variable D_i are dependent Bernoulli rvs. The dependence of the variables D_i is caused by the fact that the noise terms $\sum_{i=1}^{K} \mathbf{z}_j g_{c,ji}$ in (2.14) generally have common terms. In particular, if two columns i and j of the generator matrix $\mathbf{G}_c$ have at least a 1 in the same row, then the decoding indicators D_i and D_j are correlated. This complicates the evaluation of bounds on the frame unavailability probability (2.20).

2.4.2 Dependency Graph and Chromatic Number of a Linear Code

To capture the correlation among the indicator variables D_i, we introduce here the notion of the *dependency graph* and its chromatic number for a linear code. These appear to be novel properties of a linear code, and we will argue below that they determine the performance of the network function virtualization code $\mathcal{C}_c$ for the application at hand.

Definition 6. *Let $\boldsymbol{G} \in \mathbb{F}_2^{K' \times N'}$ be a generator matrix of a linear code. The dependency graph $\mathcal{G}(\boldsymbol{G}) = (\mathcal{V}, \mathcal{E})$ comprises a set $\mathcal{V}$ of N' vertices and a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of edges, where edge $(i, j) \in \mathcal{E}$ is included if both the ith and jth columns of $\boldsymbol{G}$ have at least a* 1 *in the same row.*

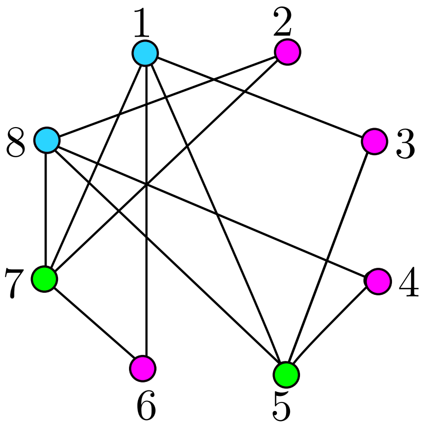


Figure 2.3 Dependency graph associated with the (8,4) network function virtualization code $\mathcal{C}_c$ in Example 1.

Example 1. *For an* $(8,4)$ *network function virtualization code* $\mathcal{C}_c$ *with the following generator matrix*

$$\boldsymbol{G}_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{2.21}$$

the resulting dependency graph $\mathcal{G}(\boldsymbol{G}_c)$ *is shown in Figure 2.3.*

The chromatic number $\mathcal{X}(\mathbf{G})$ of the graph $\mathcal{G}(\mathbf{G})$ will play an important role in the analysis. We recall that the chromatic number is the smallest number of colors needed to color the vertices of $\mathcal{G}(\mathbf{G})$, such that no two adjacent vertices share the same color (see the example in Figure 2.3). Generally, finding the chromatic number of a graph is NP-hard [91]. However, a simple upper bound on $\mathcal{X}(\mathbf{G})$ is given as [20]

$$\mathcal{X}(\mathbf{G}) \leq \Delta(\mathbf{G}) + 1, \tag{2.22}$$

where $\Delta(\mathbf{G})$ is the maximum degree of a graph $\mathcal{G}(\mathbf{G})$. A consequence of (2.22) is the following.

Lemma 1. *Let $\boldsymbol{G}$ be a $K' \times N'$ matrix, where α_r and α_c are the maximum Hamming weights of the rows and columns in $\boldsymbol{G}$, respectively. Then the chromatic number of the corresponding dependency graph $\mathcal{G}(\boldsymbol{G})$ is upper bounded as*

$$\mathcal{X}(\boldsymbol{G}) \leq \min\{N, \alpha_c(\alpha_r - 1) + 1\}. \tag{2.23}$$

Proof. According to Definition 6, we have the upper bound $\Delta(\mathbf{G}) \leq \alpha_c(\alpha_r - 1)$ and hence (2.23) follows directly from (2.22). □

2.4.3 Large Deviation Upper Bound

In this subsection, we derive an upper bound on the frame unavailability probability. The bound is based on the large deviation result in [50] for the tail probabilities of rvs $X = \sum_{i=1}^{M} X_i$, where the rvs X_i are generally dependent. We refer to this bound as the large deviation bound (LDB). The correlation of rvs $\{X_i\}$ is described in [50] by a dependency graph. This is defined as any graph $\mathcal{G}(X)$ with X_i as vertices, such that, if a vertex $i \in \{1, \ldots, M\} \backslash \{i\}$ is not connected to any vertex in a subset $\mathcal{J} \subset \{1, \ldots, M\}$, then X_i is independent of $\{X_j\}_{j \in \mathcal{J}}$.

Lemma 2 ([50])**.** *Let $X = \sum_{i=1}^{M} X_i$, where $X_i \sim$ Bern(p_i) and $p_i \in (0, 1)$ are generally dependent. For any $b \geq 0$, such that the inequality $X_i - \mathbb{E}(X_i) \geq -b$ holds for all $i \in \{1, \ldots, M\}$ with probability one, and for any $\tau \geq 0$ we have*

$$\Pr[X \leq \mathbb{E}(X) - \tau] \leq \exp\left(-\frac{S}{b^2 \mathcal{X}(\mathcal{G}(X))}\, \varphi\!\left(\frac{4b\tau}{5S}\right)\right), \tag{2.24}$$

where $S \triangleq \sum_{i=1}^{N}$ Var(X_i) and $\varphi(x) \triangleq (1 + x)\ln(1 + x) - x$. The same bound (2.24) *holds for* $\Pr(X \geq \mathbb{E}(X) + \tau)$*, where $X_i - \mathbb{E}(X_i) \leq b$ with probability one.*

The following theorem uses Lemma 2 to derive a bound on the frame unavailability probability.

Theorem 3. *Let* $\mathrm{P}_{n,k}^{\min} = \min_i\{\mathrm{P}_{n,k}(\gamma_i)\}_{i=1}^N$. *For all*

$$t \geq F^{-1}\left(\frac{N - d_{\min}}{N - \sum_{i=1}^N \mathrm{P}_{n,k}(\gamma_i)}\right), \tag{2.25}$$

the frame unavailability probability is upper bounded by

$$\mathrm{P}_u(t) \leq \exp\left(-\frac{S(t)}{b^2(t)\mathcal{X}(\boldsymbol{G}_c)}\,\varphi\left(\frac{4b(t)\left(NF(t) - F(t)\sum_{i=1}^N \mathrm{P}_{n,k}(\gamma_i) - N + d_{\min}\right)}{5S(t)}\right)\right), \tag{2.26}$$

where $b(t) \triangleq F(t)\left(1 - \mathrm{P}_{n,k}^{\min}\right)$ *and*

$$S(t) \triangleq \sum_{i=1}^N F(t)\left(1 - \mathrm{P}_{n,k}(\gamma_i)\right)\left(1 - F(t)(1 - \mathrm{P}_{n,k}(\gamma_i))\right). \tag{2.27}$$

The upper bound (2.26) *on the frame unavailability probability captures the dependency of the frame unavailability probability on both the channel and the network function virtualization code. In particular, the bound is an increasing function of the error probabilities* $\mathrm{P}_{n,k}(\gamma_i)$, *which depend on both codes. It also depends on the network function virtualization code through parameters* $d_{\min}$ *and* $\mathcal{X}(\boldsymbol{G}_c)$.

Proof. Let $X_i(t) \triangleq C_i(t)D_i$ and $X(t) = \sum_{i=1}^N X_i(t)$, where $X_i(t)$ are dependent Bernoulli rvs with probability $\mathbb{E}[X_i(t)] = \Pr[X_i(t) = 1] = F(t)\left(1 - \mathrm{P}_{n,k}(\gamma_i)\right)$. It can be seen that a valid dependency graph $\mathcal{G}(X)$ for the variables $\{X_i\}$ is the dependency graph $\mathcal{G}(\mathbf{G}_c)$ defined above. This is due to the fact that, as discussed in Section 2.4.3, the rvs X_i and X_j are dependent if and only if the ith and jth column of $\mathbf{G}_c$ have at least a 1 in a common row. We can hence apply Lemma 2 for every time t by selecting $\tau = \mathbb{E}(X) - N + d_{\min}$, and $b(t)$ as defined above. Note that this choice of $b(t)$ meets the constraint for b in Lemma 2. For $1/\mu_1 = 0$, (2.25) can be simplified as follows:

$$t \geq n\left(a - \frac{1}{\mu}\ln\left(\frac{d_{\min} - \sum_{i=1}^N \mathrm{P}_{n,k}(\gamma_i)}{N - \sum_{i=1}^N \mathrm{P}_{n,k}(\gamma_i)}\right)\right). \tag{2.28}$$

□

Remark 1. *When $t \to \infty$, we have the limit $\lim_{t\to\infty} F(t) = 1$, which implies that eventually all servers complete decoding. Letting $d^{\max} \triangleq \max\{d_i\}_{i=1}^{N}$ and $\gamma \triangleq \boldsymbol{Q}^{d^{\max}}(1,2)$, the first row and second column's entry of the matrix $\boldsymbol{Q}^{d^{\max}}$, the bound* (2.26) *reduces to*

$$\lim_{t\to\infty} \mathrm{P}_u(t) \leq \exp\left(\frac{-N\mathrm{P}_{n,k}(\gamma)}{(1-\mathrm{P}_{n,k}(\gamma))\mathcal{X}(\boldsymbol{G}_c)}\varphi\left(\frac{4\left(d_{\min}/N - \mathrm{P}_{n,k}(\gamma)\right)}{5\mathrm{P}_{n,k}(\gamma)}\right)\right). \tag{2.29}$$

This expression demonstrates the dependence of the frame unavailability probability bound (2.26) *on the number of servers N, the decoding error probability $\mathrm{P}_{n,k}(\gamma)$ for each server, the chromatic number $\mathcal{X}(\boldsymbol{G}_c)$, and minimum distance $\mathrm{d}_{\min}$ of the network function virtualization code. In particular, it can be seen that the frame unavailability probability upper bound* (2.29) *is a decreasing function of $\mathrm{d}_{\min}$, while it increases with the chromatic number, $\mathrm{P}_{n,k}(\gamma)$ and with $d^{\max}$.*

2.4.4 Union Bound

As indicated in Theorem 3, the large deviation based bound in (2.30) is only valid for large enough t, as can be observed from (2.28). Furthermore, it may generally not be tight, since it neglects the independence of the indicator variables C_i. In this subsection, a generally tighter but more complex union bound (UB) is derived that is valid for all times t.

Theorem 4. *For any subset $\mathcal{A} \subseteq \{1, \ldots, N\}$, define*

$$\mathrm{P}_{n,k}^{\min(\mathcal{A})} \triangleq \min\{\mathrm{P}_{n,k}(\gamma_i)\}_{i\in\mathcal{A}} \quad \text{and} \quad \mathrm{P}_{n,k}^{\mathcal{A}} \triangleq \sum_{i\in\mathcal{A}} \mathrm{P}_{n,k}(\gamma_i),$$

and let $\boldsymbol{G}_{\mathcal{A}}$ be the $K \times |\mathcal{A}|$, submatrix of $\boldsymbol{G}_c$, with column indices in the subset $\mathcal{A}$. Then, the frame unavailability probability $\mathrm{P}_u(t)$ is upper bounded by

$$1-\frac{1}{\binom{N}{l}}\sum_{l=N-d_{\min}+1}^{N} a_l(t)\sum_{\substack{\mathcal{A}\subseteq\{1,\ldots,N\}:\\|\mathcal{A}|=l}}\left(1-\exp\left(-\frac{S_{\mathcal{A}}}{b_{\mathcal{A}}^2\mathcal{X}(\boldsymbol{G}_{\mathcal{A}})}\varphi\left(\frac{4b_{\mathcal{A}}\left(l-N+d_{\min}-\mathrm{P}_{n,k}^{\mathcal{A}}\right)}{5S_{\mathcal{A}}}\right)\right)\right). \tag{2.30}$$

where $S_{\mathcal{A}} \triangleq \sum_{i\in\mathcal{A}} \mathrm{P}_{n,k}(\gamma_i)\left(1-\mathrm{P}_{n,k}(\gamma_i)\right)$ and $b_{\mathcal{A}} \triangleq 1-\mathrm{P}_{n,k}^{\min(\mathcal{A})}$.

Proof. Let $I_i = 1 - D_i$ be the indicator variable which equals 1 if Server i fails decoding. Accordingly, we have $I_i \sim \mathrm{Bern}(\mathrm{P}_{n,k}(\gamma_i))$. For each subset $\mathcal{A} \subseteq \{1,\ldots,N\}$, let $I_{\mathcal{A}} = \sum_{i\in\mathcal{A}} I_i$. The complement of the frame unavailability probability $\mathrm{P}_s(t) = 1-\mathrm{P}_u(t)$ can hence be written as

$$\mathrm{P}_s(t) = \Pr\left[\sum_{i=1}^{N} C_i(t) D_i > N - d_{\min}\right] \tag{2.31}$$

$$= \frac{1}{\binom{N}{l}}\sum_{l=N-d_{\min}+1}^{N} a_l(t)\sum_{\substack{\mathcal{A}\subseteq\{1,\ldots,N\}:\\|\mathcal{A}|=l}} \cdot \sum_{j=N-d_{\min}+1}^{l} \Pr\begin{bmatrix} j \text{ servers from } \mathcal{A} \text{ decode successfully} \\ \text{and} \\ l{-}j \text{ servers from } \mathcal{A} \text{ fail to decode}\end{bmatrix}$$

$$= \frac{1}{\binom{N}{l}}\sum_{l=N-d_{\min}+1}^{N} a_l(t)\sum_{\substack{\mathcal{A}\subseteq\{1,\ldots,N\}:\\|\mathcal{A}|=l}} (1-\Pr[I_{\mathcal{A}} \geq l-N+d_{min}]). \tag{2.32}$$

We can now apply Lemma 2 to the probability in (2.32) by noting that $\mathcal{G}(\mathbf{G}_{\mathcal{A}})$ is a valid dependency graph for the variables $\{I_i\}$, $i \in \mathcal{A}$. In particular, we apply Lemma 2 by setting $\tau_{\mathcal{A}} = l - N + d_{\min} - \mathbb{E}(I_{\mathcal{A}})$, $b_{\mathcal{A}} \geq I_i - \mathbb{E}[I_i]$, and $S_{\mathcal{A}} = \sum_{i\in\mathcal{A}} \mathrm{Var}\ (I_i)$, leading to

$$\Pr\left[I_{\mathcal{A}} \geq l-N+d_{\min}\right] \quad \leq \quad \exp\left(-\frac{S_{\mathcal{A}}}{b_{\mathcal{A}}^2\mathcal{X}(\mathbf{G}_{\mathcal{A}})}\varphi\left(\frac{4b_{\mathcal{A}}\left(l-N+d_{\min}-\mathrm{P}_{n,k}^{\mathcal{A}}\right)}{5S_{\mathcal{A}}}\right)\right). \tag{2.33}$$

By substituting (2.33) into (2.32), the proof is completed. □

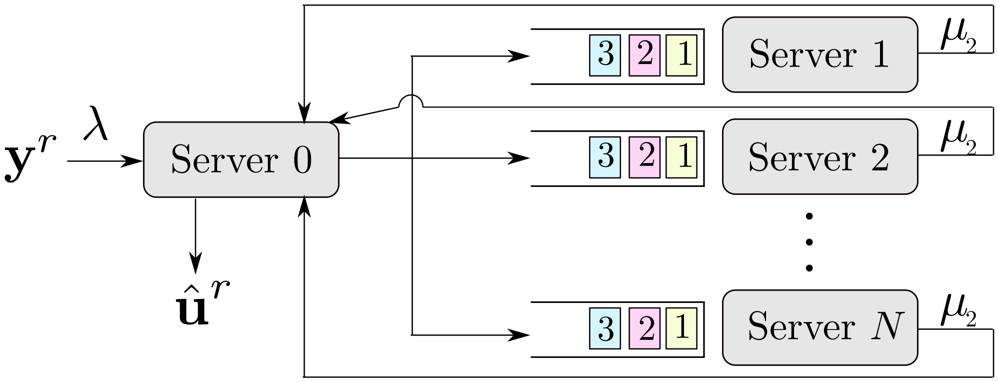


Figure 2.4 In the model studied in Section 2.5, frames arrive at the receiver according to a Poisson process with parameter λ. Server 0 in the cloud encodes the received frames using an network function virtualization code and forwards the encoded packets to servers $1, \ldots, N$ for decoding.

2.5 Random Arrivals and Queuing

In this section, we extend our analysis from one to multiple frames transmitted by the users. To this end, we study the system illustrated in Figure 2.4 with random frame arrival times and queueing at the servers. We specifically focus on the analysis of the trade-off between average latency and FER.

2.5.1 System Model

As illustrated in Figure 2.4, we assume that the arrival times of the received frames are random and distributed according to a Poisson process with a rate of λ frames per second. Upon arrival, Server 0 applies an network function virtualization code to any received frame $\mathbf{y}^r$ for $r = 1, 2, \ldots$, as described in Section II and sends each resulting coded packet $\tilde{\mathbf{y}}_i^r$ to Server i, for $i = 1, \ldots, N$. At Server i, each packet $\tilde{\mathbf{y}}_i^r$ enters a first-come-first-serve queue. After arriving at the head of the queue, each packet $\tilde{\mathbf{y}}_i^r$ requires a random time T_i to be decoded by Server i. Here, we assume that T_i is distributed according to an exponential distribution in (2.16) with an average processing time of $1/\mu_2$ per bit. Furthermore, the average time to process a frame of n bits is denoted as $1/\mu$. Also, the random variables T_i are i.i.d. across servers.

If the network function virtualization code has minimum distance $d_{\min}$, as soon as $N-d_{\min}+1$ servers decode successfully their respective packets derived from frame $\mathbf{y}^r$, the information frame $\mathbf{u}^r$ can be decoded at Server 0. We denote as T the average overall latency for decoding frame $\mathbf{u}^r$, which includes both queuing and processing.

Using (2.19), (2.20) and the fact that all servers complete decoding almost surely as $t \to \infty$, that is $C_i(t) \to 1$ as $t \to \infty$, the FER is equal to

$$\mathrm{P}_e = \Pr\left[\sum_{i=1}^{N} I_i \geq d_{\min}\right], \tag{2.34}$$

where I_i is the indicator variable that equals 1 if decoding at Server i fails. This probability can be upper bounded by the following corollary of Theorem 3.

Corollary 3. *The FER defined in* (2.34) *is upper bounded by*

$$\mathrm{P}_e \leq \exp\left(\frac{-S}{b^2\mathcal{X}(\boldsymbol{G}_{\mathcal{C}})}\varphi\left(\frac{4b\Big(d_{\min}-\sum_{i=1}^{N}\mathrm{P}_{n,k}(\gamma_i)\Big)}{5S}\right)\right), \tag{2.35}$$

where $S \triangleq \sum_{i=1}^{N}\mathrm{P}_{n,k}(\gamma_i)\left(1-\mathrm{P}_{n,k}(\gamma_i)\right)$ *and* $b \triangleq 1-\mathrm{P}_{n,k}^{\min}$.

Proof. The result follows from Theorem 3 by selecting $\tau = d_{\min} - \sum_{i=1}^{N}\mathrm{P}_{n,k}(\gamma_i)$. □

We now discuss the computation of the average delay T for different queueing management policies.

2.5.2 Per-Frame Decoding

We first study the system under a queue management policy whereby only one frame $\mathbf{y}^r$ is decoded at any time. Therefore, all servers wait until at least $N-d_{\min}+1$ servers have completed decoding of their respective packets $\tilde{\mathbf{y}}_i^r$ before moving to the next frame $r+1$, if this is currently available in the queues. Furthermore, as soon as Server 0 decodes a frame, the corresponding packets still being present in the servers' queues are evicted.

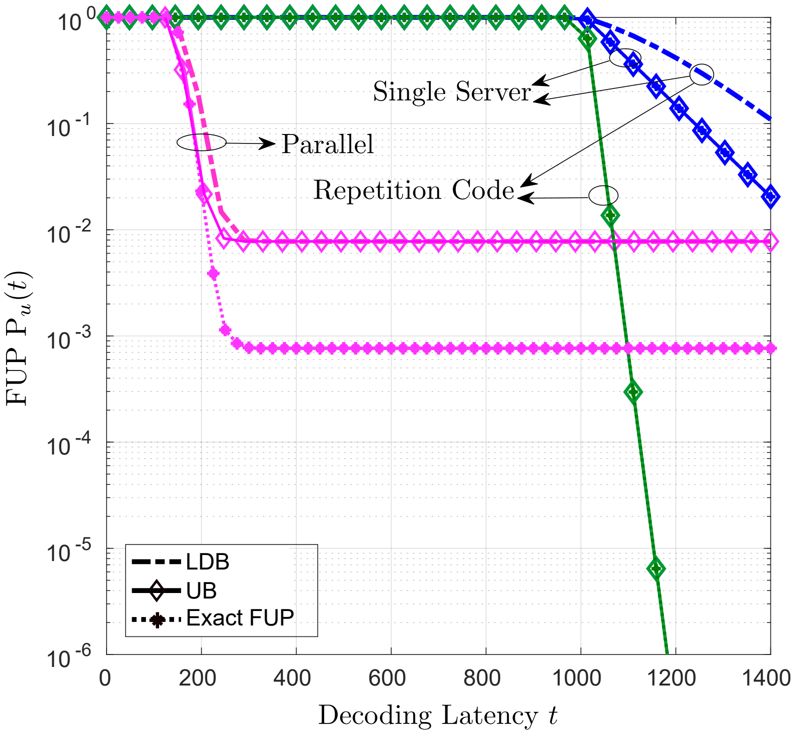


Figure 2.5 Decoding latency versus frame unavailability probability (FUP) for $L = 504, N = 8, 1/\mu_1 = 0, \mu_2 = 10, a = 1, \delta = 0.01, r = 0.5)$: (a) LDB, UB and Exact frame unavailability probability for the parallel, single-server, and repetition coding.

As a result, the overall system can be described an M/G/1 queue with arrival time λ and service time distributed according to the $(N - d_{\min} + 1)$th order statistic of the exponential distribution [54]. The latter has the pdf [89]

$$f_{T_{N-d_{\min}+1:N}}(t) = \frac{N!}{(N - d_{\min})!(d_{\min} - l)!} f_T(t) F_T(t)^{N-d_{\min}} (1 - F_T(t))^{d_{\min}-1}, \quad (2.36)$$

where $F_T(t)$ and $f_T(t)$ are the cdf and pdf of rv T_i, respectively. This queueing system was also studied in the context of distributed storage systems.

Using the Pollaczek-Khinchin formula [103], the average delay T_{pfd} of an M/G/1 queue can be obtained as

$$\frac{n(H_N - H_{d_{\min}-1})}{(N - d_{\min} + 1)\mu} + \frac{\lambda n^2[(H_N - H_{d_{\min}-1})^2 + (H_{N^2} - H_{(d_{\min}-1)^2})]}{2(N - d_{\min} + 1)^2 \mu^2 [1 - \lambda n \mu^{-1} (N - d_{\min} + 1)^{-1} (H_N - H_{d_{\min}-1})]}, \quad (2.37)$$

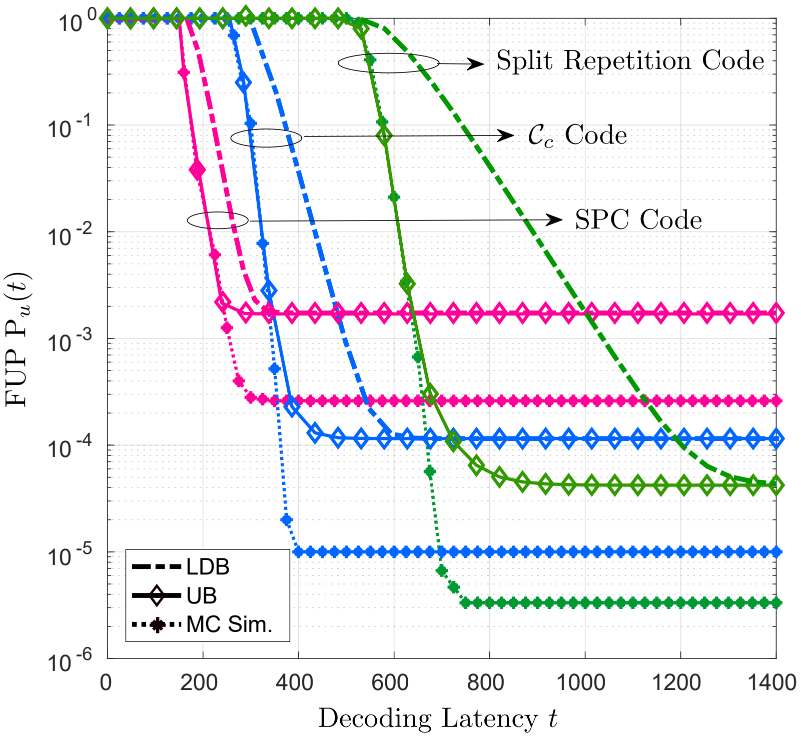


Figure 2.6 Decoding latency versus frame unavailability probability (FUP) for $L = 504, N = 8, 1/\mu_1 = 0, \mu_2 = 10, a = 1, \delta = 0.01, r = 0.5)$: LDB, UB and Monte Carlo simulation ("MC Sim.") results for split repetition code, SPC code, and the network function virtualization code $\mathcal{C}_c$ defined in (2.21).

where H_N and H_{N^2} are generalized harmonic numbers, defined by $H_N = \sum_{i=1}^{N} \frac{1}{i}$ and $H_{N^2} = \sum_{i=1}^{N} \frac{1}{i^2}$ [54]. Note that the queue is stable, and hence the average delay (2.37) is finite, if the inequality $n\lambda(H_N - H_{d_{\min}-1}) < \mu(N - d_{\min} + 1)$ holds. We refer to the described queue management scheme as per-frame decoding (pfd). This set-up is equivalent to the fork-join system studied in [54].

2.5.3 Continuous Decoding

As an alternative queue management policy, as soon as any Server i decodes its packet $\tilde{\mathbf{y}}_i^r$, it starts decoding the next packet $\tilde{\mathbf{y}}_i^{r+1}$ in its queue, if this is currently available. Furthermore, as above, as soon as Server 0 decodes a frame $\mathbf{y}^r$, all corresponding packets $\tilde{\mathbf{y}}_i^r$ still in the servers' queues are evicted. We refer this queue management policy as continuous decoding (cd).

The average delay in Equation (2.37) of per-frame decoding is an upper bound for the average delay of continuous decoding, i.e., we have $T_{\text{cd}} \leq T_{\text{pfd}}$ [54]. This is because, with per-frame decoding, all N servers are blocked until $N - d_{\min} + 1$ servers decode their designed packets. We evaluate the performance of continuous decoding using Monte Carlo methods in the next section.

2.6 Simulation Results

In this section, we provide numerical results to provide additional insights into the performance trade-off for the system shown in Figure 2.1. We first consider individual frame transmission as studied in Section 3.2 and Section 2.4, and then we study random arrivals as investigated in Section 2.5.

2.6.1 Single Frame Transmission

We first consider single frame transmission. The main goals are to validate the usefulness of the two bounds presented in Theorems 1 and 2 as design tools and to assess the importance of coding in obtaining desirable trade-offs between decoding latency and frame unavailability probability. We employ a frame length of $L = 504$ and $N = 8$ servers. The user code $\mathcal{C}_u$ is selected to be a randomly designed $(3, 6)$ regular (Gallager-type) LDPC code with $r = 0.5$, which is decoded via belief propagation.

We compare the performance of the following solutions: (i) *Standard single-server decoding*, whereby we assume, as a benchmark, the use of a single server, that is $N = 1$, that decodes the entire frame $(K = 1)$; (ii) *Repetition coding*, whereby the entire frame $(K = 1)$ is replicated at all servers; (iii) *Parallel processing*or *uncoded*, whereby the frame is divided into $K = N$ disjoint parts processed by different servers;

(iv) *Split repetition coding*, whereby the frame is split into two parts, which are each replicated at $N/2$ servers. The code has hence $K = 2$, $d_{\min} = N/2$, $\mathcal{X}(\mathbf{G}_c) = N/2$, which can be thought of as an intermediate choice between repetition coding and the parallel scheme; (v) *Single parity check code (SPC)*, with $N = K + 1$, whereby, in addition to the servers used by parallel decoding, an additional server decodes the binary sum of all other K received packets; and (vi) a network function virtualization code $\mathcal{C}_c$ with the generator matrix $\mathbf{G}_c$ defined in (2.21), which is characterized by $K = 4$. Note that, with both single-server decoding and repetition coding, we have a blocklength of $n = 1008$ for the channel code. Single-server decoding is trivially characterized by $\mathcal{X}(\mathbf{G}_c) = d_{\min} = 1$, while repetition coding is such that the equalities $\mathcal{X}(\mathbf{G}_c) = d_{\min} = 8$ hold. Furthermore, the parallel approach is characterized by $n = 126$, $d_{\min} = 1$ and $\mathcal{X}(\mathbf{G}_c) = 1$; the split repetition code is characterized by $n = 504, d_{\min} = 4$ and $\mathcal{X}(\mathbf{G}_c) = 4$; the SPC code has $n = 144, d_{\min} = 2$ and $\mathcal{X}(\mathbf{G}_c) = 2$; and the network function virtualization code $\mathcal{C}_c$ has $n = 252$, $d_{\min} = 3$ and $\mathcal{X}(\mathbf{G}_c) = 3$. The exact frame unavailability probability for a given function $\mathrm{P}_{n,k}(\cdot)$ can easily be computed for cases (i)-(iii). In particular, for single server decoding, the frame unavailability probability equals

$$\mathrm{P}_u(t) = 1 - a_1(t)(1 - \mathrm{P}_{L/r,L}(\delta)); \tag{2.38}$$

for the repetition code, the frame unavailability probability is

$$\mathrm{P}_u(t) = 1 - \sum_{i=1}^{N} a_i(t)(1 - \mathrm{P}_{L/r,L}(\delta)); \tag{2.39}$$

and for the parallel approach, we have

$$\mathrm{P}_u(t) = 1 - a_N(t)(1 - \mathrm{P}_{L/(rN),L/N}(\delta))^N. \tag{2.40}$$

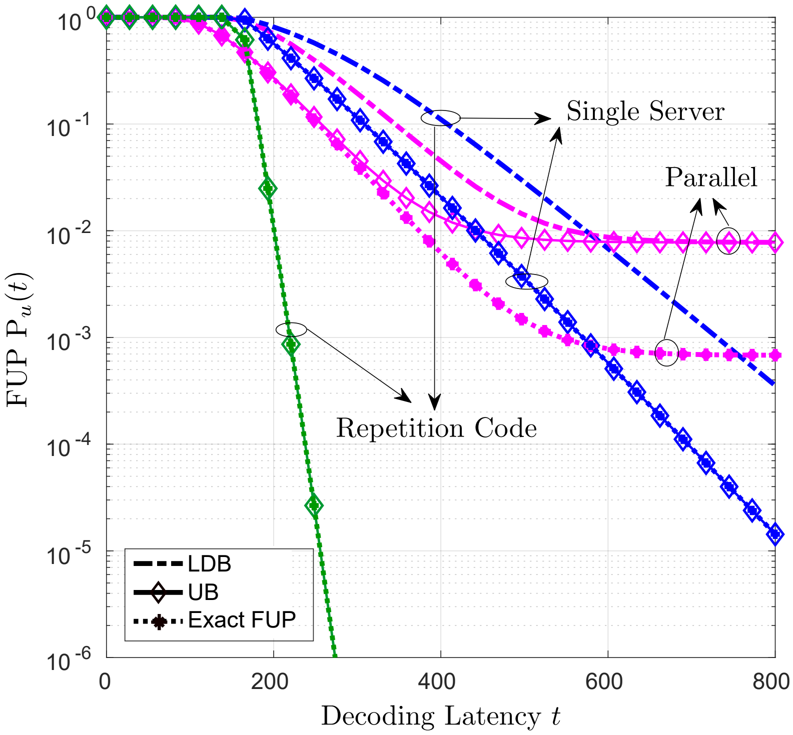


Figure 2.7 Parallel, single server and repetition code.

In contrast, the exact frame unavailability probabilities for the SPC and code $\mathcal{C}_c$ are difficult to compute, due to the discussed correlation among the decoding outcomes at the servers.

Figure 2.5 shows decoding latency versus frame unavailability probability for the LDB in Theorem 3, the UB in Theorem 4, and the exact error (2.38), (2.39), (2.40), for the first three schemes (i)-(iii), and Figure 2.6 shows the LDB in Theorem 3, the UB in Theorem 4, as well as Monte Carlo simulation results for schemes (iv), (v), and (vi). Here, we assume that the latency contribution that, is independent of the workload, is negligible, i.e., $1/\mu_1 = 0$. We also set $a = 1$ and $\mu_2 = 10$. As a first observation, Figures 2.7 and 2.8 confirms that the UB bound is tighter than the LDB.

Leveraging multiple servers in parallel for decoding is seen to yield significant gains in terms of the trade-off between latency and frame unavailability probability

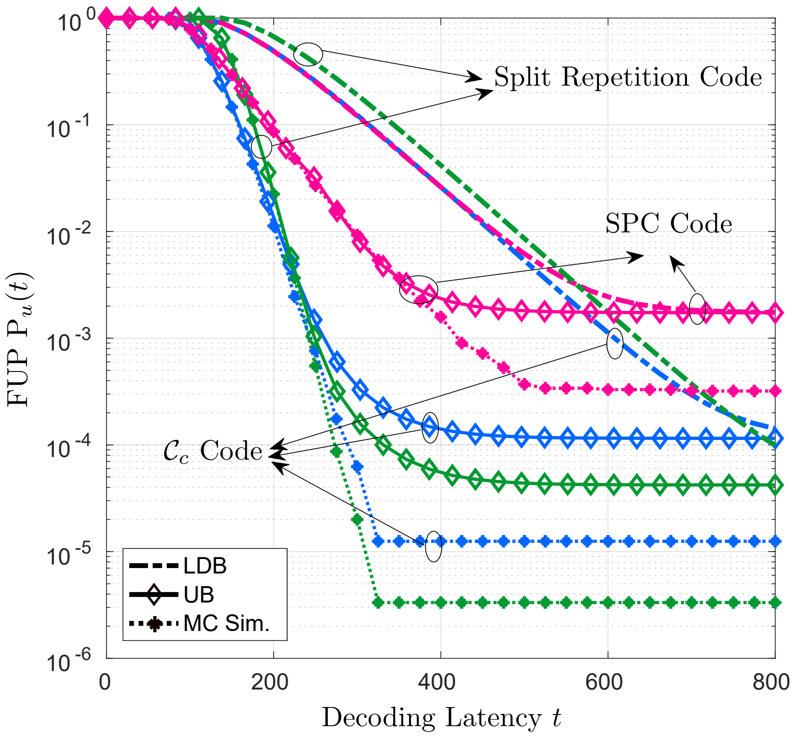


Figure 2.8 Decoding latency versus frame unavailability probability (FUP) for $(L = 504, N = 8, 1/\mu_1 = 50, \mu_2 = 20, a = 0.1, \delta = 0.01, r = 0.5)$: (a) LDB, UB and Exact frame unavailability probability for the parallel, single-server, and repetition coding; (b) LDB, UB and Monte Carlo simulation ("MC Sim.") results for split repetition code, SPC code, and the network function virtualization code $\mathcal{C}_c$ defined in (2.21). Split repetition code, SPC code and $\mathcal{C}_c$ code.

as argued also in [87] by using experimental results. In particular, the parallel scheme is observed to be preferred for lower latencies. This is due to the shorter blocklength n, which entails a smaller average decoding latency. However, the error floor of the parallel scheme is large due to the higher error probability for short blocklengths. In this case, other forms of network function virtualization coding are beneficial. To elaborate, repetition coding requires a larger latency in order to obtain acceptable frame unavailability probability performance owing to the larger blocklength n, but it achieves a significantly lower error floor. For intermediate latencies, the SPC code, and at larger latencies also both the network function virtualization code $\mathcal{C}_c$, and the split repetition code provide a lower frame unavailability probability. This demonstrates the effectiveness of network function virtualization encoding in obtaining a desirable trade-off between latency and frame unavailability probability.

In order to validate the conclusion obtained using the bounds, Figures 2.7 and 2.8 also shows the exact frame unavailability probability for the schemes (*i*)-(*iii*), as well as Monte Carlo simulation results for schemes (*iv*)-(*vi*), respectively. While the absolute numerical values of the bounds in Figures 2.5 and 2.6 are not uniformly tight with respect to the actual performance, the relative performance of the coding schemes are well matched by the analytical bounds. This provides evidence of the usefulness of the derived bounds as a tool for code design in network function virtualization systems.

Figures 2.7 and 2.8 are obtained in the same way as Figures 2.5 and 2.6, except for the parameters $\mu_1 = 0.02$, $\mu_2 = 20$, and $a = 0.1$. Unlike Figures 2.5 and 2.6, here latency may be dominated by effects that are independent of the blocklength n since we have $1/\mu_1 > 0$. The key difference with respect to Figures 2.7 and 2.8 is that, for this choice of parameters, repetition coding tends to outperform both the parallel case, and the network function virtualization code $\mathcal{C}_c$, apart from very small latencies. This is because repetition coding has the maximum resilience to the unavailability of

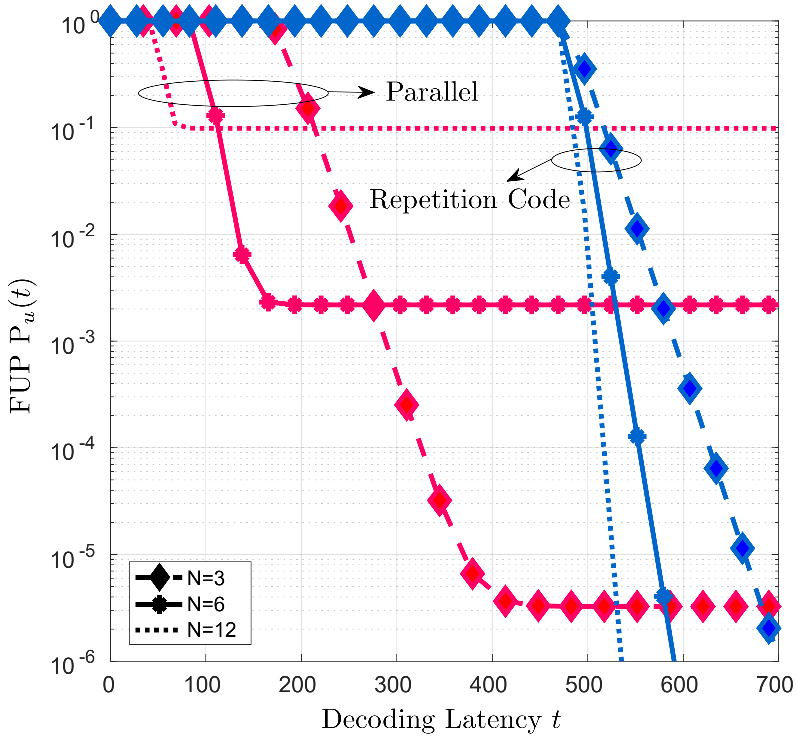


Figure 2.9 Decoding latency versus exact frame unavailability probability (FUP) for parallel and repetition coding for different number of servers $N \in \{3, 6, 12\}$ and $(L = 240, 1/\mu_1 = 0, \mu_2 = 10, a = 1, \delta = 0.03, r = 0.5)$

the servers, while not being excessively penalized by the larger blocklength n. This is not the case, however, for very small latency levels, where the network function virtualization code $\mathcal{C}_c$ provides the smallest frame unavailability probability given its shorter blocklength as compared to repetition coding and its larger $d_{\min}$, with respect to the parallel scheme.

Figure 2.9 shows the exact frame unavailability probability for the extreme cases of parallel and repetition coding for different number of servers $N \in \{3, 6, 12\}$. The figure confirms that, for both schemes, the latency decreases for a larger number of servers N. However, by increasing N, the error floor of the parallel scheme grows due to the higher channel error probability for shorter block lengths.

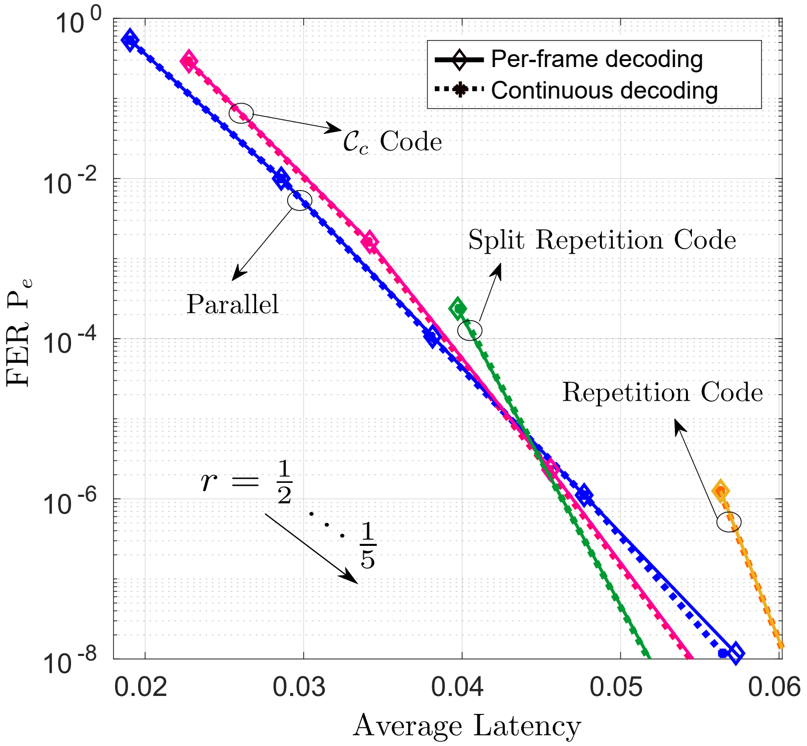


Figure 2.10 Average latency versus FER with different values of the user code rate r and for different coding schemes when the system is lightly loaded, with $L = 112, N = 8, \delta = 0.03$, $\lambda = 0.1, \mu = 500$.

2.6.2 Random Frame Transmission

We now consider the queueing system described in Section 2.5, and present numerical results that provide insights into the performance of both per-frame and continuous decoding in terms of FER versus average latency (2.34). As above, the decoding error probability is upper bounded by using [83, Theorem 33]. Both FER and average latency are a function of the user code rate r. We hence vary $r \in \{1/2, \ldots, 1/5\}$ to parametrize a trade-off curve between FER and latency. We assume a frame length of $L = 112$ bits with $N = 8$ servers, and adopt the same user code $\mathcal{C}_c$ as in the previous subsection. The average delay T_{pfd} is computed from (2.37), and T_{cd} is obtained via Monte Carlo simulations.

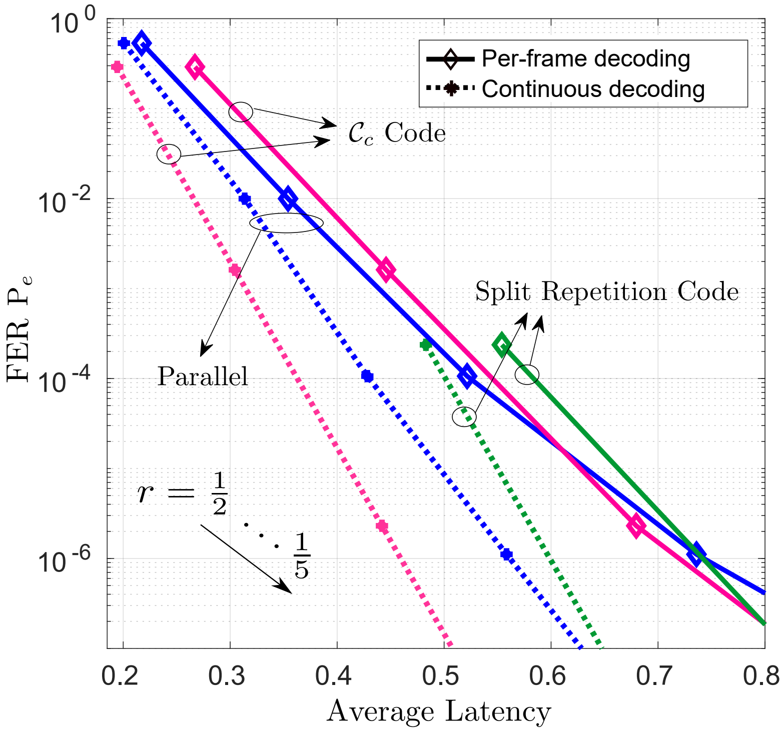


Figure 2.11 Average latency versus FER with different values of the user code rate r and for different coding schemes when the system is heavily loaded, with $L = 112, N = 8, \delta = 0.03,\ \lambda = 1, \mu = 50$.

Figures 2.10 and 2.11 compare the performance of repetition coding, the network function virtualization code $\mathcal{C}_c$ with the generator matrix (2.21), and the parallel approach as defined above. Figure 2.10 considers a lightly loaded system with $\lambda = 0.1$ frames per second and $\mu = 500$ frames per second, while Figure 2.11 shows a highly loaded system with both $\lambda = 1$ frames per second and $\mu = 50$ frames per second.

First, by comparing the two figures we observe that per-frame decoding and continuous decoding have a similar performance when the system is lightly loaded (see Figure 2.10), while continuous decoding yields a smaller average latency than per-frame decoding when the system is heavily loaded (see Figure 2.11). This is because, in the former case, it is likely that a frame is decoded successfully before the next one arrives. This is in contrast to heavily loaded systems in which the average latency becomes dominated by queuing delays. We also note that, for repetition coding, the performance of per-frame decoding and continuous decoding coincides in both lightly or heavily loaded systems, since decoding is complete as soon as one server decodes successfully.

Also, by comparing the performance of different codes, we recover some of the main insights obtained from the study of the isolated frame transmission. In particular, the parallel approach outperforms all other schemes for low average delays due to its shorter block length n. In contrast, repetition coding outperforms all other schemes in FER for large average delay because of its large block length n and consequently low probability of decoding error (not shown). Furthermore, we observe that split repetition coding is to be preferred for small values of FER.

Finally, Figure 2.12 demonstrates the behavior of the average latency as the arrival rate λ increases and the system becomes more heavily loaded. We observe that, for a lightly loaded system, the latencies of per frame and continuous decoding are similar, while continuous decoding is preferable for a large number of λ. This is because per-frame decoding requires all servers to wait until at least $N - d_{\min} + 1$

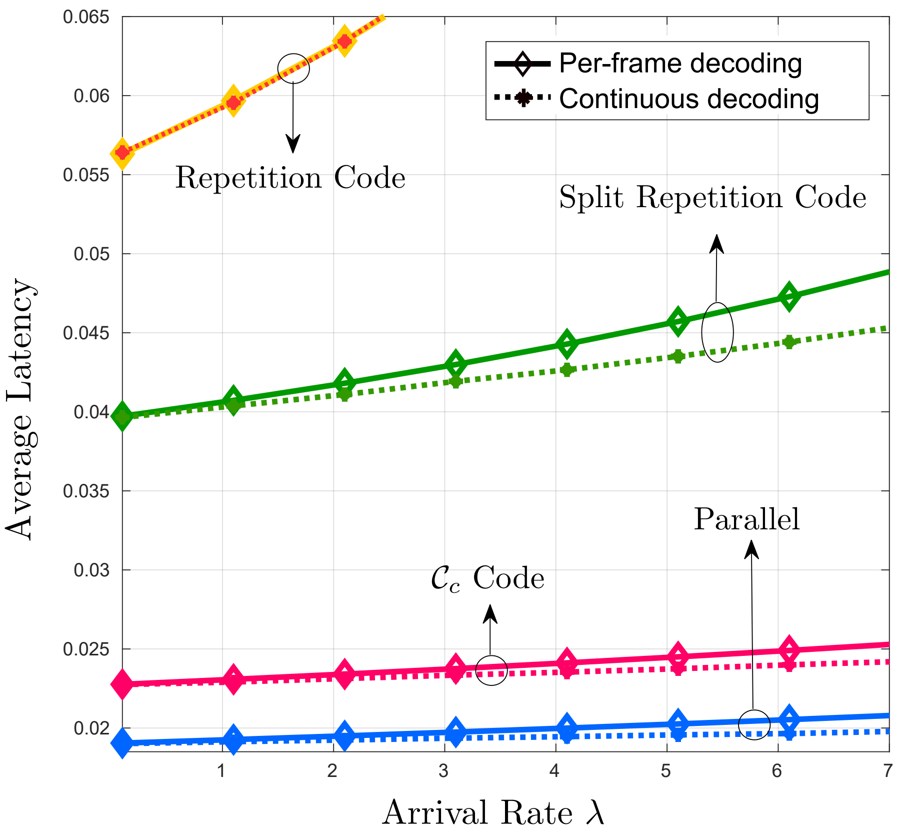


Figure 2.12 Average latency versus arrival rate λ ($L = 112, N = 8, r = 0.5, \mu = 500$).

servers have completed decoding of their respective packets before moving on to the next frame.

2.7 Discussion and Concluding Remarks

We obtained the performance of a novel coded NFV approach for the uplink of a C-RAN system in which decoding takes place at a multi-server or multi-core cloud processor. This approach is based on the linear combination of the received packets prior to their distribution to the servers or cores, and on the exploitation of the algebraic properties of linear channel codes. The method can be thought of as an application of the emerging principle of coded computing to NFV. Analysis and simulation results demonstrate the significant gains that linear coding of received packets, or NFV coding, can yield in terms of trade-off between decoding latency and FER. Among interesting open problems, we mention here the design of optimal NFV codes and the extension of the principle of NFV coding to Gaussian channels.

The source of the error probability in the system is due to first, decoding noisy packets in servers and second a random respond time that each server needs to decode. Furthermore, we explore the tradeoff between effective parameters of the NFV system such as minimum distance and chromatic number of the code. These analysis direct us to interesting future design of NFV codes considering the application of the principle of coded NFV.

CHAPTER 3

PRIVATE AND SECURE MATRIX MULTIPLICATION WITH FLEXIBLE COMMUNICATION LOAD

3.1 Introduction

3.1.1 Motivation and Problem Definition

At the core of many signal processing and machine learning applications are tensor operations, most notably large matrix multiplications [51]. In the presence of practically sized data sets, such operations are typically carried out using distributed computing platforms with a master server and multiple workers that can operate in parallel over distinct parts of the data set. The master server plays the role of the parameter server, distributing data to the workers and periodically reconciling their internal state [69]. Workers are commercial off-the-shelf servers that are characterized by possible temporary failures and delays [28].

Straggling workers can affect the computation latency by orders of magnitude, e.g., [55, 106]. While current distributed computing platforms conventionally handle straggling servers by means of replication of computing tasks [49], recent work has shown that encoding the input data can help reduce the computation latency. More generally, coding is able to control the trade-off between computational delay and communication load between workers and master server [66, 111, 74, 6, 7, 37, 35, 39, 38, 98]. Furthermore, stochastic coding can help keeping both input and output data secure from the workers, assuming that the latter are honest, i.e., carrying out the prescribed protocol, but curious [81, 113, 22, 56, 107, 32, 27, 80]. This chapter contributes to this line of work by investigating the trade-off between computational delay and communication load as a function of the privacy level.

As illustrated in Figures 3.1 and 3.2, we focus on the basic problem of computing a matrix multiplication $\mathbf{C} = \mathbf{AB}$ in a distributed computing system of P workers that can process each only a fraction $1/m$ and $1/n$ of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. In the first setup under study, illustrated in Figure 3.1, both matrices $\mathbf{A}$ and $\mathbf{B}$ are to be kept private from the workers. Here, three performance criteria are of interest:

- the recovery threshold P_R, that is, the number of workers that need to complete their task before the master server can recover the product $\mathbf{C}$;
- the communication load C_L between workers and master server, i.e., the amount of information to be downloaded from the workers;
- the maximum number P_C of colluding servers that ensures perfect secrecy for both data matrices $\mathbf{A}$ and $\mathbf{B}$.

In the second setup of interest shown in Figure 3.2, only matrix $\mathbf{A}$ is private, while matrix $\mathbf{B}$ is selected from a public data set $\mathcal{B}$. In this case, apart from the security constraint on $\mathbf{A}$, we only impose a privacy constraint on the identity of the specific matrix $\mathbf{B} \in \mathcal{B}$ of interest. As a motivation for this second setup, consider a recommender system based on collaborative filtering [85]. In this case, recommendations are based on the product of two matrices, one describing the profile of a user, or a group of users, and one representing features of the items of interest, such as movies, music or TV shows. The users' profile matrix can be modelled by the private matrix $\mathbf{A}$. Hence ensuring the privacy of users' data; while the items' data matrix for each category is represented by one of the matrices in the public data set $\mathcal{B} = \{\mathbf{B}^{(k)}\}_{k=1}^{L}$. This latter assumption captures the constraint that users may want to keep the confidential types of items they are interested in. For this problem, the criteria of interest are still P_R and P_C, and we simplify the problem by setting $P_C = 1$. This chapter focuses on the design of coding and computing techniques for both problems.

3.1.2 Related Work

In order to put our contribution in perspective, we briefly review prior related work. Consider first solutions that provide no security guarantees, i.e., $P_C = 0$, for the problem in Figure 3.1. As a direct extension of [66], a first approach is to use product codes that apply separately the maximum distance separable (MDS) codes to encode the two matrices [68]. The recovery threshold of this scheme is improved by [111], which introduces *polynomial codes.* The construction in [111] is proved to be optimal under the assumption that *minimal communication* is allowed between workers and master server. MatDot codes are introduced in [39], resulting in a lower recovery threshold at the expense of a larger communication load. The construction in [37] bridges the gap between polynomial and MatDot codes and presents PolyDot codes, yielding a trade-off between recovery threshold and communication load. An extension of this scheme, termed Generalized PolyDot (GPD) codes improves on the recovery threshold of PolyDot codes [35], which is independently obtained also by the construction in [112]. GPD codes in [35] are used to design a unified coded computing strategy for the training of deep neural networks.

Much less work has been done in the literature for the case in which security constraints are factored in, i.e., where $P_C \neq 0$, for the problem of Figure 3.1. In [113], Lagrange coding is presented that achieves the minimum recovery threshold for multilinear functions by generalizing MatDot codes. In [81, 80], coded schemes have been used to develop multi-party computation techniques to calculate arbitrary polynomials of massive matrices, preserving the security of the data matrices. In [22, 56, 32] a reduction of the communication load is obtained by extending polynomial codes. While these works focus on either minimizing recovery threshold or communication load, the *trade-off* between these two fundamental quantities has not been addressed in the open literature to the best of our knowledge. A new class of secure distributed matrix multiplication and its capacity is studied in [53].

In the second part of this work, we study a connection between secure matrix multiplication and private information retrieval (PIR), as illustrated in Figure 3.2. The private information retrieval problem was introduced in [26] and has been widely studied in recent years, e.g., in [43, 110, 99, 14, 41, 59, 58, 61, 23, 100]. In [61] and [23] the private information retrieval setup was investigated for the problem of distributed matrix multiplication illustrated in Figure 3.2 that imposes private information retrieval guarantees for the index of matrix $\mathbf{B}$ within a public library. In [61], a coding strategy is proposed that combines the private information retrieval scheme for non-colluding servers (i.e., with $P_C = 1$) [26] with polynomial codes [111]. In [23], the authors introduce a related approach for this problem, and show that it outperforms the scheme proposed in [61] in terms of upload and download cost. The code design in [23] focuses on the minimization of the communication load, and does not explore the trade-off between this metric and the recovery threshold.

3.1.3 Main Contribution

In this dissertation, we first present a novel class of secure computation codes, referred to as secure GPD (SGPD) codes, for the setup in Figure 3.1, SGPD codes generalize GPD codes to operate at a flexible communication load level. This yields a new achievable trade-off between recovery threshold P_R and communication load C_L as a function of a prescribed number of colluding workers P_C. In the process, we also introduce a novel perspective on distributed computing codes based on the signal processing concepts of convolution and z-transform. SGPD codes. Then, SGPD codes are modified to offer a solution, introduced here for the first time, for the scenario in Figure 3.2. This is done through concatenation with the private information retrieval code in [61], which ensures both secrecy of the input matrix $\mathbf{A}$ and privacy of the identity for the desired matrix in the library $\mathcal{B}$ if $P_C = 1$. The resulting codes are referred to as private and secure GPD (PSGPD) codes. They generalize the approach in [23], enabling a trade-off between (upload) communication load and recovery threshold. We finally illustrate the benefits of the proposed codes, which offer a flexible trade-off between communication load and recovery threshold, by analyzing the overall completion time due to both computation and communication.

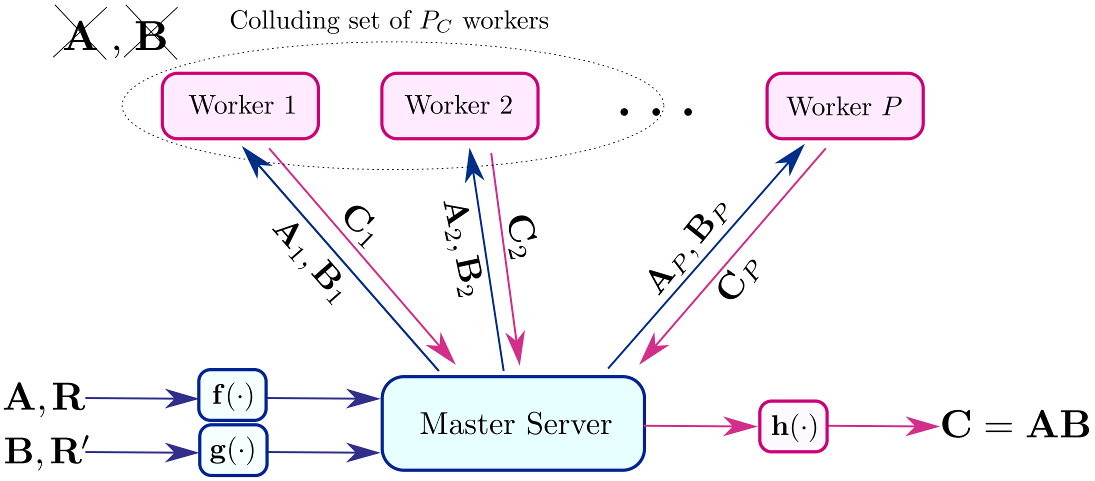
3.1.4 Organization

The rest of the chapter is organized as follows. In Section 3.2, we present the system models for secure matrix multiplication (Figure 3.1 in Section 3.2.3) and for private and secure matrix multiplication (Figure 3.2 in Section 3.2.4), respectively. In Section 3.3, we propose an intuitive interpretation of the GPD code introduced in [39]. Using z-transforms, Section 3.4 proposes a novel extension of GPD codes by imposing a security constraint on the data matrices and deriving the resulting trade-off between recovery threshold P_R and communication load C_L. In this section, we also study overall completion latency encompassing both computation and communication latencies for SGPD codes. In Section 3.5, we address the setup in Figure 3.2, again with respect to the trade-off between P_R and C_L and to the overall completion latency. This chapter is concluded in Section 3.6.

3.2 Problem Statement

3.2.1 Notation

Throughout this chapter, we denote a matrix with upper boldface letters (e.g., $\mathbf{X}$), and lower boldface letters indicate a vector or a sequence of matrices (e.g., $\mathbf{x}$). Furthermore, a math calligraphic font refers to a set (e.g., $\mathcal{X}$). A set $\mathbb{F}$ represents the Galois field with cardinality $|\mathbb{F}|$. We denote by $\mathbb{N}$ the set of all non-zero positive integers, and for some $a, b \in \mathbb{N}$, $a \leq b$, $[a, b] \triangleq \{a, a+1, \ldots, b\}$. For any real number r, $\lceil r \rceil$ represents the largest integer nearest to r. The function $H(\cdot)$ represents the entropy of its argument, and $I(X;Y)$ denotes the mutual information of the random variables X and Y.



$\cancel{\mathbf{A}}, \cancel{\mathbf{B}}$
Colluding set of P_C workers
Worker 1
Worker 2
Worker P
$\mathbf{A}_1, \mathbf{B}_1$
$\mathbf{C}_1$
$\mathbf{A}_2, \mathbf{B}_2$
$\mathbf{C}_2$
$\mathbf{A}_P, \mathbf{B}_P$
$\mathbf{C}_P$
$\mathbf{A}, \mathbf{R}$
$\mathbf{B}, \mathbf{R}'$
$\mathbf{f}(\cdot)$
$\mathbf{g}(\cdot)$
Master Server
$\mathbf{h}(\cdot)$
$\mathbf{C} = \mathbf{AB}$

Figure 3.1 Secure matrix multiplication: the master server encodes both input matrices $\mathbf{A}$ and $\mathbf{B}$, to be kept secure from the workers, and both random matrices $\mathbf{R}$ and $\mathbf{R}'$, respectively, to define the computational tasks of the slave servers or workers. The workers may fail or straggle, and they are honest but curious, with colluding subsets of workers of size at most P_C. The master server must be able to decode the product $\mathbf{C} = \mathbf{AB}$ from the output of a subset of P_R servers, which defines the recovery threshold.

3.2.2 System Model

As illustrated in Figures 3.1 and 3.2, we consider a distributed computing system with a master server and P slave servers or workers. The master server is interested in computing securely the matrix product $\mathbf{C} = \mathbf{AB}$ of two data matrices $\mathbf{A}$ and $\mathbf{B}$ with dimensions $T \times S$ and $S \times D$, respectively. The matrices have i.i.d. uniformly distributed entries from a sufficient large finite field $\mathbb{F}$, with $|\mathbb{F}| > P$. More precisely, we will consider two scenarios. In the first, both matrices $\mathbf{A}$ and $\mathbf{B}$ are available at the master server and contain confidential data that should be kept secure from the workers (see Figure 3.1). In the second, only matrix $\mathbf{A}$ contains confidential information, and there are L public matrices in the set $\mathcal{B} = \{\mathbf{B}^{(r)}\}_{r=1}^{L}$ from which the master node wishes to compute the product $\mathbf{C}^{(\kappa)} = \mathbf{AB}^{(\kappa)}$ for some κth index $\kappa \in [1, L]$. The index must be kept private against the workers (see Figure 3.2). In the subsequent sections, we first describe the system model for the setup in Figure

3.1, referred to as *secure matrix multiplication*, followed by the setup for the model in Figure 3.2, referred to as *private and secure matrix multiplication.*

3.2.3 Secure Matrix Multiplication

For the scenario in Figure 3.1 workers receive information on matrices $\mathbf{A} \in \mathbb{F}^{T \times S}$ and $\mathbf{B} \in \mathbb{F}^{S \times D}$ from the master server; they process this information and they respond to the master server, which finally recovers the product $\mathbf{C} = \mathbf{AB}$ with minimal computational effort. Due to communication and complexity constraints, each worker can receive only TS/m and SD/n symbols, respectively, for some integers m and n. The workers are honest but curious. Accordingly, we impose the secrecy constraint that, even if up to $P_C < P$ workers collude, the workers cannot obtain any information about both matrices $\mathbf{A}$ and $\mathbf{B}$ based on the data received from the master server.

To keep the data secure and to leverage possible computational redundancy at the workers (namely, if $P/m > 1$ and/or $P/n > 1$), the master server sends encoded versions of the input matrices to the workers due to the above mentioned communication and complexity constraints. Specifically, it produces the encoded matrices $\mathbf{A}_p = \mathbf{f}_p(\mathbf{A}, \mathbf{R})$, where $\mathbf{R}$ is a random matrix of dimension $T' \times S'$, for some integers T' and S' to be defined below, via the function

$$\mathbf{f}_p : \mathbb{F}^{T \times S} \times \mathbb{F}^{T' \times S'} \to \mathbb{F}^{T/t \times S/s}, \tag{3.1}$$

for some integers t and s such that $m = st$. The resulting TS/m entries in the output of function $\mathbf{f}_p$ are then sent to worker p, with $p \in [1, P]$. Likewise, the master server computes the encoded matrices $\mathbf{B}_p = \mathbf{g}_p(\mathbf{B}, \mathbf{R}')$, where $\mathbf{R}'$ is a random matrix of dimension $S' \times D'$, for some integers S' and D' to be defined below, using the function

$$\mathbf{g}_p : \mathbb{F}^{S \times D} \times \mathbb{F}^{S' \times D'} \to \mathbb{F}^{S/s \times D/d}, \tag{3.2}$$

for some integers s and d such that $n = sd$. The resulting SD/n entries in $\mathbf{B}_p$ are then sent to worker p. The random matrices $\mathbf{R}$ and $\mathbf{R}'$ consists of i.i.d. uniformly distributed entries from a field $\mathbb{F}$. The security constraint imposes the condition

$$I(\mathbf{A}_{\mathcal{P}}, \mathbf{B}_{\mathcal{P}}; \mathbf{A}, \mathbf{B}) = 0, \tag{3.3}$$

for all subsets of $\mathcal{P} \subset [1, P]$ of P_C workers, where the random matrices $\mathbf{R}$ and $\mathbf{R}'$ serve as random keys in order to meet the security constraint (3.3) [94].

Each worker p computes the product $\mathbf{C}_p = \mathbf{A}_p\mathbf{B}_p$ of the encoded sub-matrices $\mathbf{A}_p$ and $\mathbf{B}_p$. The master server collects a subset of $P_R \leq P$ outputs from the workers as defined by the subset $\{\mathbf{C}_p\}_{p\in\mathcal{P}_R}$ with $|\mathcal{P}_R| = P_R$. It then applies a decoding function as $\mathbf{h}\left(\{\mathbf{C}_p\}_{p\in\mathcal{P}_R}\right)$,

$$\mathbf{h} : \underbrace{\mathbb{F}^{T/t\times D/d} \times \cdots \times \mathbb{F}^{T/t\times D/d}}_{P_R \text{ times}} \to \mathbb{F}^{T\times D}. \tag{3.4}$$

Note that *correct decoding* translates into the condition

$$H(\mathbf{AB}|\{\mathbf{C}_p\}_{p\in\mathcal{P}_R}) = 0. \tag{3.5}$$

A coding and decoding strategy that satisfies condition (3.3) and (3.5) is said to be *feasible.*

For given parameters m and n the performance of a coding and decoding scheme is measured by the triple (P_C, P_R, C_L), where C_L is defined as

$$C_L = \sum_{p\in\mathcal{P}_R} |\mathbf{C}_p|; \tag{3.6}$$

$|\mathbf{C}_p|$ is the dimension of the product matrix $\mathbf{C}_p$ computed by worker p. Note that condition (3.5) requires the inequality $\min\{P_R/m, P_R/n\} \geq 1$ or $P_R \geq P_{R,\min} \triangleq \max\{m, n\}$, which is hence a lower bound for the minimum recovery threshold. Furthermore, the communication load is lower bounded by $C_L \geq C_{L,\min} \triangleq TD$, which is the size of the product $\mathbf{C} = \mathbf{AB}$.

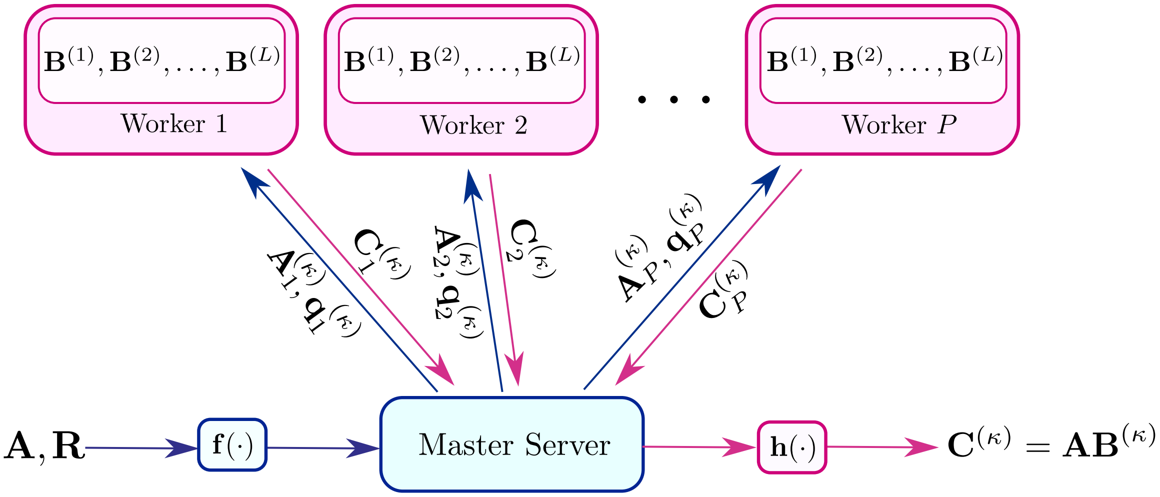

$\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(L)}$
Worker 1
$\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(L)}$
Worker 2
$\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(L)}$
Worker P
$\mathbf{A}_1^{(\kappa)}, \mathbf{q}_1^{(\kappa)}$
$\mathbf{C}_1^{(\kappa)}$
$\mathbf{A}_2^{(\kappa)}, \mathbf{q}_2^{(\kappa)}$
$\mathbf{C}_2^{(\kappa)}$
$\mathbf{A}_P^{(\kappa)}, \mathbf{q}_P^{(\kappa)}$
$\mathbf{C}_P^{(\kappa)}$
$\mathbf{A}, \mathbf{R}$
$\mathbf{f}(\cdot)$
Master Server
$\mathbf{h}(\cdot)$
$\mathbf{C}^{(\kappa)} = \mathbf{A}\mathbf{B}^{(\kappa)}$

Figure 3.2 Private and secure matrix multiplication: the master server encodes the input matrix $\mathbf{A}$, to be kept secret from the workers, and generates the encoded matrix $\mathbf{A}_p^{(\kappa)}$ for each worker p. It also sends a query $\mathbf{q}_p^{(\kappa)}$ as a function of the index $\kappa \in [1, L]$, to be kept private from workers, of the desired product $\mathbf{C}^{(\kappa)} = \mathbf{A}\mathbf{B}^{(\kappa)}$, with matrices $\{\mathbf{B}^{(r)}\}_{r=1}^{L}$ available at all workers. The non-colluding workers may fail or straggle, and they are honest but curious. The master server must be able to decode the product $\mathbf{C}^{(\kappa)}$ from the output of a subset of P_R servers, which defines the recovery threshold.

3.2.4 Private and Secure Matrix Multiplication

In this subsection, we discuss the private and secure matrix multiplication problem illustrated in Figure 3.2. In this setup, the master server wishes to compute the product $\mathbf{C}^{(\kappa)} = \mathbf{A}\mathbf{B}^{(\kappa)}$ of a confidential input matrix $\mathbf{A}$ with a matrix $\mathbf{B}^{(\kappa)}$ from a set of public matrices $\{\mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(L)}\}$, while keeping the index κ of the matrix $\mathbf{B}^{(\kappa)}$ of interest private from the workers.

Similar to the secure model in Figure 3.1, we consider a distributed computing system with a master server and P honest but curious workers. The master server contains a confidential data matrix $\mathbf{A}$ with dimension $T \times S$. Each worker has access to the library $\mathcal{B}$, which consists of L distinct matrices $\{\mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(L)}\}$, each with dimension $S \times D$. As above, all matrices contain data symbols chosen uniformly i.i.d. from a sufficient large finite field $\mathbb{F}$, with $|\mathbb{F}| > P$. The master server is interested in computing the matrix product $\mathbf{C}^{(\kappa)} = \mathbf{A}\mathbf{B}^{(\kappa)}$ of the data matrix $\mathbf{A}$ and of a matrix

$\mathbf{B}^{(\kappa)}$ for some index $\kappa \in [1, L]$. This should be done while keeping the data matrix $\mathbf{A}$ secret against the workers in the same sense as in the scenario of Figure 3.1, while also ensuring that the index κ is kept secret from the workers.

To do so, as in the private information retrieval problem [99, 14], the master server generates P query vectors $\mathbf{q}_1^{(\kappa)}, \ldots, \mathbf{q}_P^{(\kappa)} \in \mathbb{F}^L$, for some $L > 1$ as a function of the desired index κ and sends each worker $p \in [1, P]$, the query vector $\mathbf{q}_p^{(\kappa)}$. We assume that the workers do not collude, i.e., we set $P_C = 1$. Extensions to any $P_C > 1$ are possible and are left for future work. We note that, when the input matrix $\mathbf{A}$ is an identity matrix, the setup reduces to a private information retrieval problem.

To keep the data matrix $\mathbf{A}$ secure against workers, the master server sends each worker $p \in [1, P]$ an encoded version $\mathbf{A}_p^{(\kappa)} = \mathbf{f}_p(\kappa, \mathbf{A}, \mathbf{R}) \in \mathbb{F}^{T/t \times S/s}$ which is a function of index κ, and through it, of the query $\mathbf{q}_p^{(\kappa)}$, of the data matrix $\mathbf{A}$ and of a random matrix $\mathbf{R}$, for some integers t and s such that $m = ts$.

Upon receiving $(\mathbf{q}_p^{(\kappa)}, \mathbf{A}_p^{(\kappa)})$, each worker p uses the query $\mathbf{q}_p^{(\kappa)}$ to derive an $S/s \times D/d$ matrix $\mathbf{B}_p^{(\kappa)} = \mathbf{g}_p(\mathbf{q}_p^{(\kappa)}, \mathcal{B}) \in \mathbb{F}^{S/s \times D/d}$ from the library $\mathcal{B}$ by using an encoding function

$$\mathbf{g}_p : \mathbb{F}^L \times \underbrace{\mathbb{F}^{S \times D} \times \cdots \times \mathbb{F}^{S \times D}}_{L \text{ times}} \to \mathbb{F}^{S/s \times D/d}, \tag{3.7}$$

for some integers s and d such that $n = sd$. We emphasize that, unlike the setup considered in Figure 3.1, the *content* of the desired matrix $\mathbf{B}^{(\kappa)}$ is not secure against workers, since the library $\mathcal{B}$ is public. Each worker p then computes the product $\mathbf{C}_p^{(\kappa)} = \mathbf{A}_p^{(\kappa)} \mathbf{B}_p^{(\kappa)}$ and sends it to the master server. The master server collects a subset $\{\mathbf{C}_p^{(\kappa)}\}_{p \in \mathcal{P}_R}$ of $P_R \leq P$ outputs from the workers with $|\mathcal{P}_R| = P_R$. It then applies a decoding function $\mathbf{h}(\{\mathbf{C}_p^{(\kappa)}\}_{p \in \mathcal{P}_R})$, as in (3.4), in order to retrieve the desired product $\mathbf{C}^{(\kappa)} = \mathbf{A}\mathbf{B}^{(\kappa)}$.

To guarantee the *secrecy of input matrix* $\mathbf{A}$, in a manner similar to (3.3), we have the constraint

$$I(\mathbf{A}_p^{(\kappa)}, \mathbf{B}_p^{(\kappa)}, \mathbf{q}_p^{(\kappa)}, \mathcal{B}; \mathbf{A}) = 0, \tag{3.8}$$

for all $p \in [1, P]$. Following the private information retrieval formulation on [61], in order to ensure the *privacy of index* κ, for some value of κ the information available at each worker should be statistically indistinguishable from that available for any other value $\kappa' \neq \kappa$. Mathematically, for all $\kappa, \kappa' \in [1, L]$ with $\kappa' \neq \kappa$ and for all workers $p \in [1, P]$, we have the condition

$$(\mathbf{q}_p^{(\kappa)}, \mathbf{A}_p^{(\kappa)}, \mathbf{C}_p^{(\kappa)}, \mathcal{B}) \sim (\mathbf{q}_p^{(\kappa')}, \mathbf{A}_p^{(\kappa')}, \mathbf{C}_p^{(\kappa')}, \mathcal{B}), \tag{3.9}$$

that is, the joint distribution of variables $(\mathbf{q}_p^{(\kappa')}, \mathbf{A}_p^{(\kappa')}, \mathbf{C}_p^{(\kappa')}, \mathcal{B})$ should be the same for any pair of index values $\kappa' \neq \kappa$. Finally, the *correct decoding* requirement is defined as in (3.5), that is

$$H(\mathbf{A}\mathbf{B}^{(\kappa)} | \{\mathbf{C}_p^{(\kappa)}\}_{p \in \mathcal{P}_R}) = 0. \tag{3.10}$$

A coding and decoding strategy that satisfies conditions (3.8), (3.9), and (3.10) is said to be *feasible*. For given parameters m and n the performance is measured by the pair (P_R, C_L), with $P_C = 1$, where C_L is the communication load defined in (3.6).

3.3 Background: Generalized PolyDot Code without Security Constraint

In this section, we consider the system model shown in Figure 3.1 and review the GPD construction first proposed in [39] and later improved in [112, 35] for the special case of no secrecy constrains, i.e., $P_C = 0$. In the process, we propose a novel intuitive interpretation of GPD encoding and decoding based on the distributed computation of samples from convolutions via z-transforms.

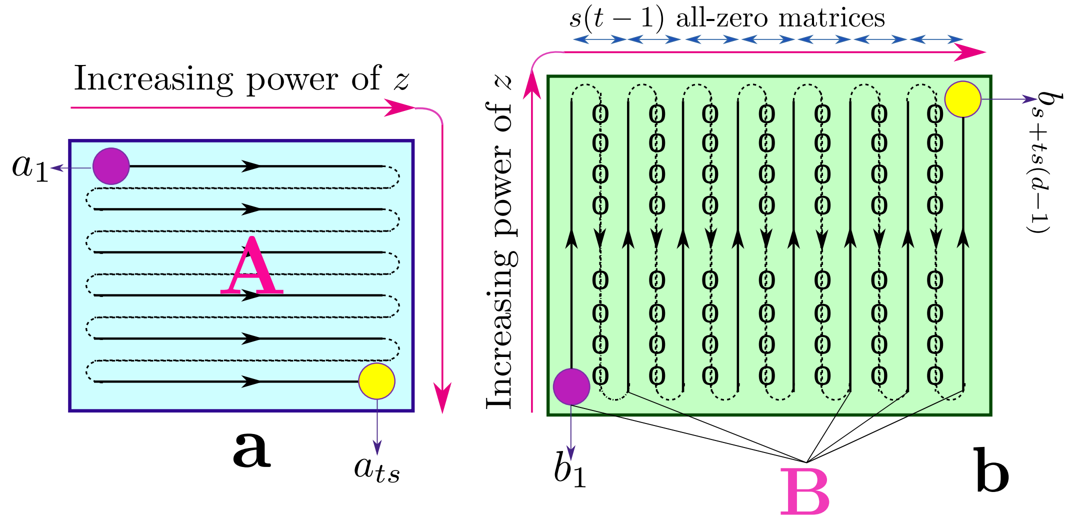


Figure 3.3 Construction of the time sequences **a** and **b** used to define the generalized PolyDot (GPD) code. The zero dashed lines in **b** indicates all-zero block sequences. Each solid arrows in **a** and **b** shows a distinct row of **A** and a column of **B**, respectively.

We start by recalling that the GPD coding scheme achieves the best currently known trade-off between recovery threshold P_R and communication load C_L for $P_C = 0$, i.e., under no security constraint. The entangled polynomial codes of [112] have the same properties in terms of (P_R, P_C). The GPD codes for $P_C = 0$ also achieve the optimal recovery threshold among all linear coding strategies in the cases of $t = 1$ or $d = 1$, also they minimize the recovery threshold for the minimum communication load $C_{L,\min}$ [111, 112].

The GPD code splits the data matrices **A** and **B** both horizontally and vertically as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \dots & \mathbf{A}_{1,s} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{t,1} & \dots & \mathbf{A}_{t,s} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \dots & \mathbf{B}_{1,d} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{s,1} & \dots & \mathbf{B}_{s,d} \end{bmatrix}. \tag{3.11}$$

The parameters s, t, and d can be set arbitrarily under the constraints $m = ts$ and $n = sd$. Note that polynomial codes set $s = 1$, while MatDot codes have $t = d = 1$

[37]. All sub-matrices $\mathbf{A}_{i,j}$ and $\mathbf{B}_{k,l}$ have dimensions $T/t \times S/s$ and $S/s \times D/d$, respectively. The GPD code computes each block (i,j) of the product $\mathbf{C} = \mathbf{AB}$, namely $\mathbf{C}_{i,j} = \sum_{k=1}^{s} \mathbf{A}_{i,k}\mathbf{B}_{k,j}$, for $i \in [1,t]$ and $j \in [1,d]$, in a distributed fashion. This is done by means of polynomial encoding and polynomial interpolation. As we review next, the computation of block $\mathbf{C}_{i,j}$ can be interpreted as the evaluation of the middle sample of the convolution $\mathbf{c}_{i,j} = \mathbf{a}_i * \mathbf{b}_j$ between the block sequences $\mathbf{a}_i = [\mathbf{A}_{i,1}, \ldots, \mathbf{A}_{i,s}]$ and $\mathbf{b}_j = [\mathbf{B}_{s,j}, \ldots, \mathbf{B}_{1,j}]$. In fact, the sth sample of the block sequence $\mathbf{c}_{i,j}$ equals $\mathbf{C}_{i,j}$, i.e., $[\mathbf{c}_{i,j}]_s = \mathbf{C}_{i,j}$. The computation is carried out distributively in the frequency domain by using z-transforms with different workers being assigned distinct samples in the frequency domain.

To elaborate, define the block sequence $\mathbf{a}$ obtained by concatenating the block sequences $\mathbf{a}_i$ as $\mathbf{a} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_t\}$. Pictorially, a sequence $\mathbf{a}$ is obtained from the matrix $\mathbf{A}$ by reading the blocks in the left-to-right top-to-bottom order, as seen in Figure 3.3. We also introduce the longer time block sequence $\mathbf{b}$ as

$$\mathbf{b} = \{\mathbf{b}_1, \mathbf{0}, \mathbf{b}_2, \mathbf{0}, \ldots, \mathbf{b}_d\}, \tag{3.12}$$

with $\mathbf{0}$ being a block sequence of $s(t^* - 1)$ all-zero block matrices with dimensions $S/s \times D/d$. The sequence $\mathbf{b}$ can be obtained from the matrix $\mathbf{B}$ by following the bottom-to-top left-to-right order shown in Figure 3.3 and by adding the all-zero block sequences between any two columns of the matrix $\mathbf{B}$.

In the frequency domain, the z-transforms of sequences $\mathbf{a}$ and $\mathbf{b}$ are obtained as

$$\mathbf{F}_{\mathbf{a}}(z) = \sum_{r=0}^{ts-1} [\mathbf{a}]_{r+1} z^r = \sum_{i=1}^{t} \sum_{j=1}^{s} \mathbf{A}_{i,j} z^{s(i-1)+j-1}, \tag{3.13}$$

$$\mathbf{F}_{\mathbf{b}}(z) = \sum_{r=0}^{s-1+ts(d-1)} [\mathbf{b}]_{r+1} z^r = \sum_{k=1}^{s} \sum_{l=1}^{d} \mathbf{B}_{k,l} z^{s-k+ts(l-1)}, \tag{3.14}$$

respectively. The master server evaluates the polynomials $\mathbf{F}_{\mathbf{a}}(z)$ and $\mathbf{F}_{\mathbf{b}}(z)$ in P non-zero distinct points $z_1, \ldots, z_P \in \mathbb{F}$ and sends the corresponding linearly encoded matrices $\mathbf{A}_p = \mathbf{F}_{\mathbf{a}}(z_p)$ and $\mathbf{B}_p = \mathbf{F}_{\mathbf{b}}(z_p)$ to server p. The encoding functions are hence given by the polynomial evaluations (3.13) and (3.14), for $z_1, \ldots, z_p$. Server p computes the multiplication $\mathbf{F}_{\mathbf{a}}(z_p)\mathbf{F}_{\mathbf{b}}(z_p)$ and sends it to the master server. The master server computes the inverse z-transform for the received products $\{\mathbf{A}_p\mathbf{B}_p\}_{p\in\mathcal{P}_R} = \{\mathbf{F}_{\mathbf{a}}(z_p)\mathbf{F}_{\mathbf{b}}(z_p)\}_{p\in\mathcal{P}_R}$, obtaining the convolution $\mathbf{a} * \mathbf{b}$.

From the convolution $\mathbf{a} * \mathbf{b}$, we can see that the master server is able to compute all the desired blocks $\mathbf{C}_{i,j}$ by reading the middle samples of the convolutions $\mathbf{c}_{i,j} = \mathbf{a}_i * \mathbf{b}_j$ from samples of the sequence $\mathbf{c} = \mathbf{a} * \mathbf{b}$ in the order $[\mathbf{c}]_{s-1} = \mathbf{C}_{1,1}, [\mathbf{c}]_{2s-1} = \mathbf{C}_{2,1}, \ldots, [\mathbf{c}]_{ts-1} = \mathbf{C}_{t,1}, [\mathbf{c}]_{s-1+t^*s} = \mathbf{C}_{1,2}, \ldots, [\mathbf{c}]_{ts-1+t^*s} = \mathbf{C}_{t,2}, \ldots$. Note that, in particular, the zero block subsequences added to sequence $\mathbf{b}$ ensure that no interference from the other convolutions, $\mathbf{c}_{i',j'}$ affects the middle (sth) sample of a convolution $\mathbf{c}_{i,j}$ with $i' \neq i$ and $j' \neq j$.

To carry out the inverse transform, the master server needs to collect as many values $\mathbf{F}_{\mathbf{a}}(z_p)\mathbf{F}_{\mathbf{b}}(z_p)$ as there are samples of the sequence $\mathbf{a} * \mathbf{b}$, yielding the recovery threshold

$$P_R = tsd + s - 1. \tag{3.15}$$

Equivalently, in terms of the underlying polynomial interpretation, the master server needs to collect a number of evaluations of the polynomial $\mathbf{F}_{\mathbf{a}}(z)\mathbf{F}_{\mathbf{b}}(z)$ equal to the degree of $\mathbf{F}_a(z)\mathbf{F}_b(z)$ plus one. This computation is of complexity order $\mathcal{O}(TDP_R(\log(P_R))^2)$ [37]. Furthermore, the communication load is given as

$$C_L = P_R\frac{TD}{td}, \tag{3.16}$$

where $TD/(td)$ is the size of each matrix $\mathbf{F}_{\mathbf{a}}(z)\mathbf{F}_{\mathbf{b}}(z)$.

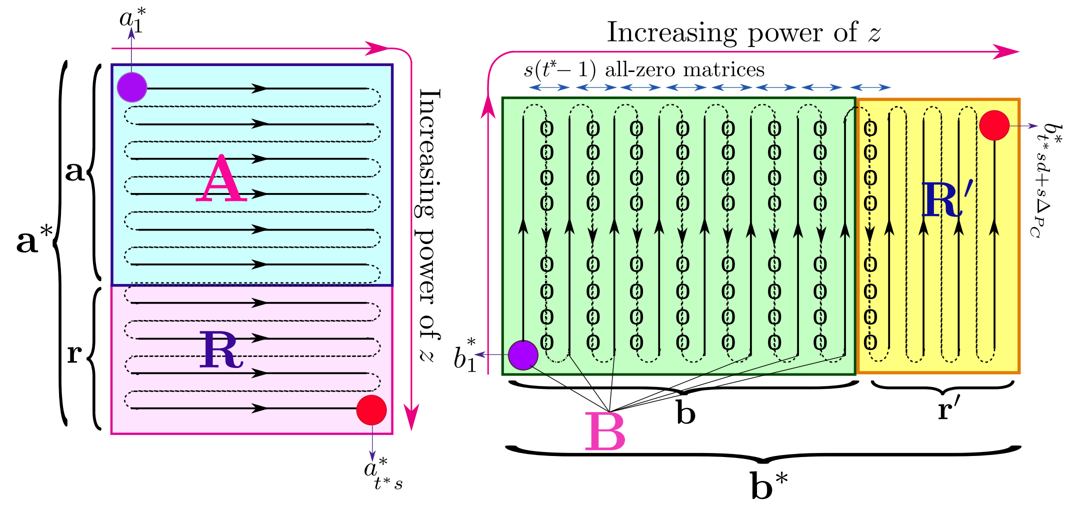


Figure 3.4 Construction of the time block sequences $\mathbf{a}^* = [\mathbf{a}, \mathbf{r}]$ and $\mathbf{b}^* = [\mathbf{b}, \mathbf{r}']$ in (3.20) and (3.21) used to define the SGPD code for the case $s < t$. The zero dashed lines in $\mathbf{b}$ and $\mathbf{r}'$ indicate all-zero block sequences.

3.4 Secure PolyDot Code

In this section, we propose a novel extension of the GPD code that is able to ensure the secrecy constraint for any $P_C < P$. We also derive the corresponding achievable set of triples (P_C, P_R, C_L). As we will discuss, the projection of this set onto the plane defined by the condition $P_C = 0$ includes the set of pairs (P_R, C_L) in (3.15) and (3.16) obtained by the GPD code [35]. The proposed secure GPD (SGPD) code augments matrices $\mathbf{A}$ and $\mathbf{B}$ by adding P_C random block matrices to the input matrices $\mathbf{A}$ and $\mathbf{B}$, in a manner similar to prior works [81, 113, 22, 56, 32], yielding augmented matrices $\mathbf{A}^*$ and $\mathbf{B}^*$. As we will see, a direct application of the GPD codes to these matrices is suboptimal.

In contrast, we propose a novel way to construct sequences $\mathbf{a}^*$ and $\mathbf{b}^*$ from matrices $\mathbf{A}^*$ and $\mathbf{B}^*$ that enables the definition of a more efficient code by means of the z-transform approach discussed in the previous section. To this end, we follow the design criterion of decreasing the recovery threshold P_R for a given communication load C_L. Based on the discussion in the previous section, this goal can be realized by decreasing the length of the sequence $\mathbf{c}^* = \mathbf{a}^* * \mathbf{b}^*$, which can in turn be ensured by reducing the length of the sequence $\mathbf{b}^*$ for a given length of the sequence $\mathbf{a}^*$. We accomplish this objective by *(i)* adaptively appending rows *or* columns with random elements to matrix $\mathbf{A}$, and, correspondingly columns *or* rows to $\mathbf{B}$, which can reduce the recovery threshold; and *(ii)* modifying the zero padding procedure (see Figure 3.3) for the construction of sequence $\mathbf{b}^*$. In order to account for point *(i)*, we consider separately the two cases $s < t$ and $s \geq t$.

3.4.1 Secure Generalized PolyDot Code: The $s < t$ Case

As illustrated in Figure 3.4, when $s < t$, we augment the input matrices $\mathbf{A}$ and $\mathbf{B}$ by adding

$$\Delta_{P_C} \triangleq \left\lceil \frac{P_C}{s} \right\rceil, \tag{3.17}$$

random row and column blocks to matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. Accordingly, the $t^* \times s$ augmented block matrix $\mathbf{A}^*$ with $t^* = t + \Delta_{P_C}$ is obtained as

$$\mathbf{A}^* = \begin{bmatrix} \mathbf{A} \\ \mathbf{R} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1,1} & \dots & \mathbf{A}_{1,s} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{t,1} & \dots & \mathbf{A}_{t,s} \\ \mathbf{R}_{1,1} & \dots & \mathbf{R}_{1,s} \\ \vdots & \ddots & \vdots \\ \mathbf{R}_{\Delta_{P_C},1} & \dots & \mathbf{R}_{\Delta_{P_C},s} \end{bmatrix}, \tag{3.18}$$

while the $s \times d^*$ augmented matrix $\mathbf{B}^* = [\mathbf{B}\ \mathbf{R}']$ with $d^* = d + \Delta_{P_C}$ is obtained as

$$\mathbf{B}^* = \begin{bmatrix} \mathbf{B}_{1,1} & \dots & \mathbf{B}_{1,d} & \mathbf{R}'_{s,1} & \dots & \mathbf{R}'_{s,\Delta_{P_C}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{s,1} & \dots & \mathbf{B}_{s,d} & \mathbf{R}'_{1,1} & \dots & \mathbf{R}'_{1,\Delta_{P_C}} \end{bmatrix}. \tag{3.19}$$

In (3.18) and (3.19), if s divides P_C, all block matrices $\mathbf{R}_{i,j} \in \mathbb{F}^{\frac{T}{t} \times \frac{S}{s}}$ and $\mathbf{R}'_{i,j} \in \mathbb{F}^{\frac{S}{s} \times \frac{D}{d}}$ are generated with i.i.d. uniform random elements in $\mathbb{F}$. Otherwise, if $\Delta_{P_C} - P_C/s > 0$, the last $s\Delta_{P_C} - P_C$ matrices in (3.18), with right-to-left ordering in the last row of $\mathbf{R}_{i,j}$, and in (3.19) with top-to-bottom ordering in the last column of $\mathbf{R}'_{i,j}$, are all-zero block matrices.

As illustrated in Figure 3.4, in the SGPD scheme, the block sequence $\mathbf{a}^*$ is defined in the same way as in the conventional GPD, yielding

$$\mathbf{a}^* = \{\mathbf{a}_1, \ldots, \mathbf{a}_t, \mathbf{r}_1, \ldots, \mathbf{r}_{\Delta_{P_C}}\}, \tag{3.20}$$

where $\mathbf{r}_i$ is the ith row of the block matrix $\mathbf{R}$, $i \in [1, \Delta_{P_C}]$. We also define the time block sequence $\mathbf{b}^* = \{\mathbf{b}, \mathbf{r}'\}$ as

$$\mathbf{b}^* = \{\mathbf{b}_1, \mathbf{0}, \mathbf{b}_2, \mathbf{0}, \ldots, \mathbf{b}_d, \mathbf{0}, \mathbf{r}'_1, \mathbf{r}'_2, \ldots, \mathbf{r}'_{\Delta_{P_C}}\}, \tag{3.21}$$

where $\mathbf{0}$ is block sequences of $s(t^* - 1)$ all-zero block matrices, respectively, with dimensions $S/s \times D/d$, while $\mathbf{r}'_j$ is the jth column of the random matrix $\mathbf{R}'$. The key novel idea of this construction is that no zero matrices are introduced between the columns of matrix $\mathbf{R}'$. As shown in Theorem 5 below, this construction allows the master server to recover all the desired submatrices $\mathbf{C}_{i,j}$ for $i \in [1, t]$ and $j \in [1, d]$ from the middle samples of the convolutions $\mathbf{c}_{i,j} = \mathbf{a}_i * \mathbf{b}_j$ (see Figure 3.5 for an illustration).

Theorem 5. *For a given security level $P_C < P$, the proposed SGPD code achieves the recovery threshold*

$$P_R = \begin{cases} tsd + s - 1, & \text{if } P_C = 0, \\ t^*s(d+1) + s\Delta_{P_C} - 1, & \text{if } P_C \geq 1 \text{ and } \Delta_{P_C} = \frac{P_C}{s}, \\ t^*s(d+1) - s\Delta_{P_C} + 2P_C - 1, & \text{if } P_C \geq 1 \text{ and } \Delta_{P_C} > \frac{P_C}{s}, \end{cases} \tag{3.22}$$

and the communication load (3.16)*, where $t^* = t + \Delta_{P_C}$ and $d^* = d + \Delta_{P_C}$ for any integer values $t, s,$ and d such that $s < t$, $m = ts$, and $n = sd$.*

Proof. The z-transform of sequences $\mathbf{a}^*$ and $\mathbf{b}^*$ are given respectively as

$$\begin{aligned}\mathbf{F}_{\mathbf{a}^*}(z) &= \underbrace{\sum_{i=1}^{t}\sum_{j=1}^{s}\mathbf{A}^*_{i,j}z^{s(i-1)+(j-1)}}_{\triangleq\ \mathbf{F}_1(z)} \\ &+ \underbrace{\sum_{i=t+1}^{t^*}\sum_{j=1}^{s}\mathbf{A}^*_{i,j}z^{s(i-1)+j-1}}_{\triangleq\ \mathbf{F}_2(z)}, \qquad (3.23)\\ \mathbf{F}_{\mathbf{b}^*}(z) &= \underbrace{\sum_{k=1}^{s}\sum_{l=1}^{d}\mathbf{B}^*_{k,l}z^{s-k+t^*s(l-1)}}_{\triangleq\ \mathbf{F}_3(z)} \\ &+ \underbrace{\sum_{k=1}^{s}\sum_{l=d+1}^{d^*}\mathbf{B}^*_{k,l}z^{t^*sd+s(l-d)-k}}_{\triangleq\ \mathbf{F}_4(z)}. \qquad (3.24)\end{aligned}$$

The master server evaluates $\mathbf{F}_{\mathbf{a}^*}(z)$ and $\mathbf{F}_{\mathbf{b}^*}(z)$ at P non-zero distinct points $z_1, \ldots, z_P \in \mathbb{F}$, which define the encoding functions, and sends both matrices $\mathbf{A}_p = \mathbf{F}_{\mathbf{a}^*}(z_p)$ and $\mathbf{B}_p = \mathbf{F}_{\mathbf{b}^*}(z_p)$ to worker p. Worker p performs the multiplication $\mathbf{F}_{\mathbf{a}^*}(z_p)\mathbf{F}_{\mathbf{b}^*}(z_p)$, and sends the results back to the master server. To reconstruct all blocks $\mathbf{C}_{i,j}$ of matrix $\mathbf{C} = \mathbf{AB}$, the master server carries out a polynomial interpolation, or equivalently, it computes the inverse z-transform, upon receiving a number of multiplication results equal to at least the length of the sequence $\mathbf{c}^* = \mathbf{a}^* * \mathbf{b}^*$. As we detail next, the (i, l) block $\mathbf{C}_{i,l} = \sum_{r=1}^{s}\mathbf{A}_{i,r}\mathbf{B}_{r,l}$, for all $i \in [1, t]$ and $l \in [1, d]$, of matrix $\mathbf{C} = \mathbf{AB}$ can be seen equal to the $(si - 1 + (l - 1)t^*s)$th sample of the convolution $\mathbf{c}^* = \mathbf{a}^* * \mathbf{b}^*$. An illustration can be found in Figure 3.5.

To see this, we first note that, by the properties of GPD codes, matrix $\mathbf{C}_{i,l}$ is the coefficient of the monomial $z^{si-1+(l-1)t^*s}$ in $\mathbf{F}_1(z)\mathbf{F}_3(z)$. Note that this holds since the polynomial $\mathbf{F}_1(z)$ and $\mathbf{F}_3(z)$ are defined as GPD codes. We now need to show that no other contribution to this term arises from the products $\mathbf{F}_1(z)\mathbf{F}_4(z)$, $\mathbf{F}_2(z)\mathbf{F}_3(z)$, and $\mathbf{F}_2(z)\mathbf{F}_4(z)$. The terms in the product $\mathbf{F}_1(z)\mathbf{F}_4(z)$ have exponents

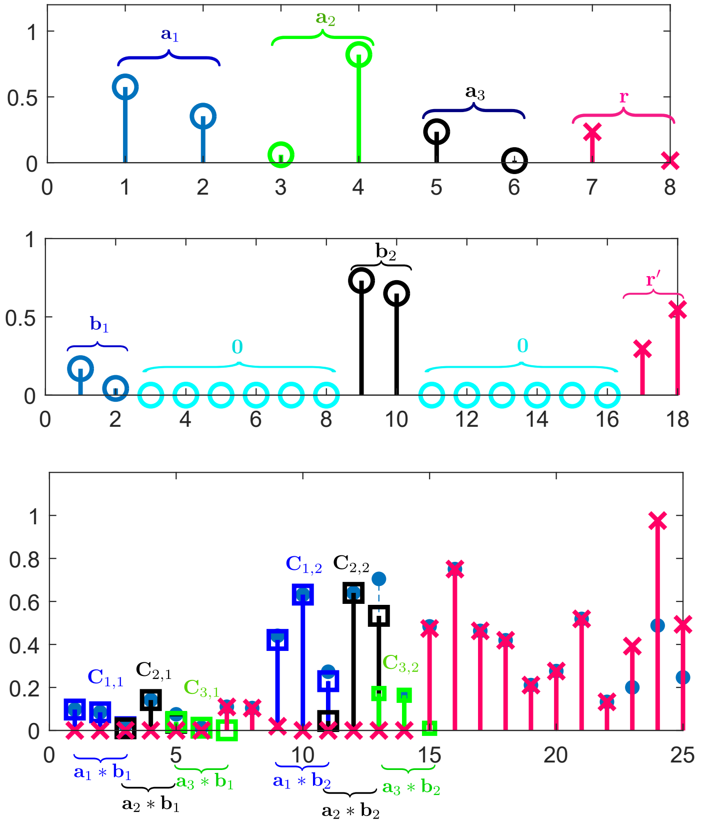


Figure 3.5 Outcome of the communication $\mathbf{C}_{i,j} = \mathbf{a}_i * \mathbf{b}_j$ for $t = 3, s = 2, d = 2$, and $P_C = 2$. Dashed blue stems with filled markers represent the convolution $\mathbf{c}^*$. Individual convolutions $\mathbf{c}_{i,j}$ are shown in different colors with square markers. Contributions from one or both random matrices are shown as red crosses. The desired submatrices $\mathbf{C}_{i,j}$ are seen to equal the corresponding samples from the sequence $\mathbf{c}^*$, associated with the center points of the individual convolutions.

$(t^*sd+s(i-1)+s(l-d)-1)$, for $i \in [1,t]$ and $l \in [d+1,d^*]$, which do not include the desired values $(si-1+(l-1)t^*s)$ for $i \in [1,t]$ and $l \in [1,d]$. A similar discussion applies to the product $\mathbf{F}_2(z)\mathbf{F}_3(z)$, whose exponents are $(s(i+t^*l-t^*)-1)$, for $i \in [t+1,t^*]$ and $l \in [1,d]$, and $\mathbf{F}_2(z)\mathbf{F}_4(z)$, whose exponents are $(t^*sd+s(i-1)+s(l-d)-1)$, for $i \in [t+1,t^*]$ and $l \in [d+1,d^*]$.

In order to recover the convolution $\mathbf{c}^*$, the master server needs to collect a number of values of the product $\mathbf{F}_\mathbf{a}(z)\mathbf{F}_\mathbf{b}(z)$ equal to the length of the sequence $\mathbf{c}^*$, which can be computed as the degree $\deg\left(\mathbf{F}_\mathbf{a}(z)\mathbf{F}_\mathbf{b}(z)\right)+1$, where $\deg(\mathbf{F}_\mathbf{a}(z)\mathbf{F}_\mathbf{b}(z))$

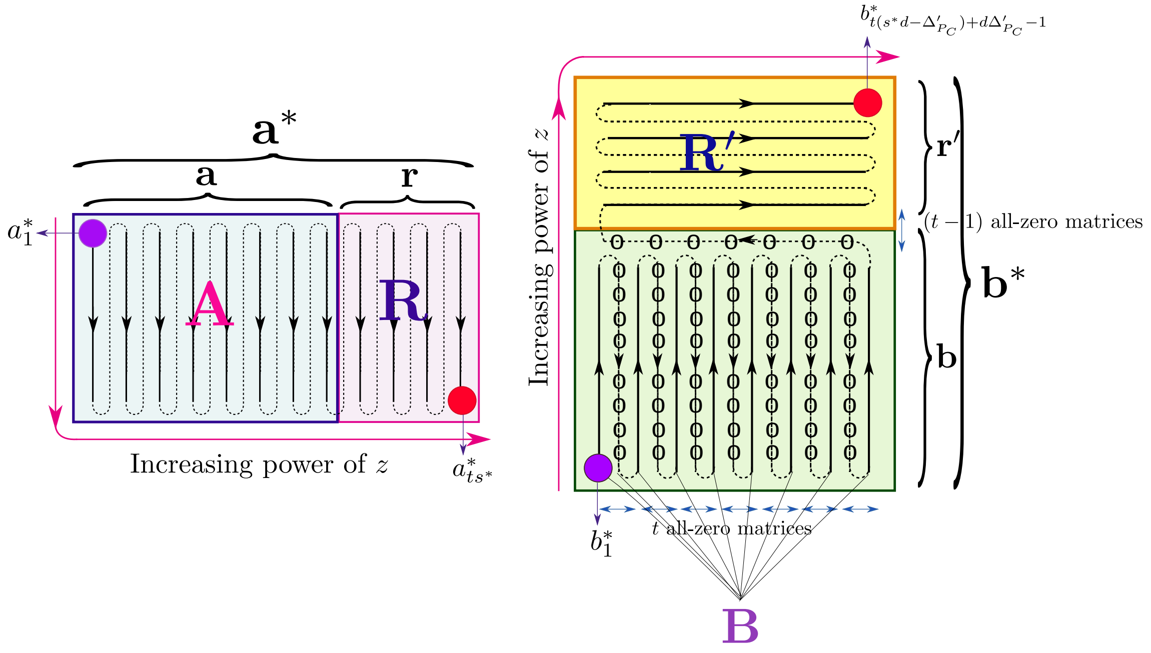


Figure 3.6 Construction of the time block sequences $\mathbf{a}^*$ and $\mathbf{b}^*$ in (3.31) and (3.32) used to define the secure generalized PolyDot (SGPD) code for the case $s \geq t$. The solid line and the zero dashed lines in $\mathbf{b}^*$ indicate columns of $\mathbf{B}$ and all-zero block sequences, respectively.

is

$$\deg(\mathbf{F}_{\mathbf{a}}(z)\mathbf{F}_{\mathbf{b}}(z)) = \begin{cases} t^*s(d+1) + s\Delta_{P_C} - 1, & \text{if } \Delta_{P_C} = \frac{P_C}{s}, \\ dst^* - s\Delta_{P_C} + 2P_C + t - 2, & \text{if } \Delta_{P_C} > \frac{P_C}{s}. \end{cases} \tag{3.25}$$

For $P_C \geq 1$ this implies the recovery threshold P_R in (3.22). The communication load C_L in (3.16) follows from the fact that there are $TD/(td)$ entries in $\mathbf{F}_{\mathbf{a}^*}(z_p)\mathbf{F}_{\mathbf{b}^*}(z_p)$, for all $p \in [1, P_R]$.

The security constraint in Equation (3.3) can be proved in a manner similar to [22] by the following steps:

$$
\begin{aligned}
&I(\mathbf{A},\mathbf{B};\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}})\\
=&H(\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}})-H(\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}}|\mathbf{A},\mathbf{B})\\
\overset{(a)}{=}&H(\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}})-H(\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}}|\mathbf{A},\mathbf{B})\\
&+H(\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}}|\mathbf{A},\mathbf{B},\mathbf{R}_1,\ldots,\mathbf{R}_{P_C},\mathbf{R}'_1,\ldots,\mathbf{R}'_{P_C})\\
=&H(\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}})-I(\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}};\mathbf{R}_1,\ldots,\mathbf{R}_{P_C},\mathbf{R}'_1,\ldots,\mathbf{R}'_{P_C}|\mathbf{A},\mathbf{B})\\
=&H(\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}})-H(\mathbf{R}_1,\ldots,\mathbf{R}_{P_C},\mathbf{R}'_1,\ldots,\mathbf{R}'_{P_C}|\mathbf{A},\mathbf{B})\\
&+H(\mathbf{R}_1,\ldots,\mathbf{R}_{P_C},\mathbf{R}'_1,\ldots,\mathbf{R}'_{P_C}|\mathbf{A},\mathbf{B},\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}})\\
\overset{(b)}{=}&H(\mathbf{A}_{\mathcal{P}},\mathbf{B}_{\mathcal{P}})-H(\mathbf{R}_1,\ldots,\mathbf{R}_{P_C},\mathbf{R}'_1,\ldots,\mathbf{R}'_{P_C})\\
\overset{(c)}{\leq}&H(\mathbf{A}_{\mathcal{P}})+H(\mathbf{B}_{\mathcal{P}})-\sum_{p=1}^{P_C}H(\mathbf{R}_p)-\sum_{p=1}^{P_C}H(\mathbf{R}'_p)\\
\overset{(d)}{=}&H(\mathbf{A}_{\mathcal{P}})+H(\mathbf{B}_{\mathcal{P}})-P_C\frac{TS}{m}\log|\mathbb{F}|-P_C\frac{SD}{n}\log|\mathbb{F}|\\
\overset{(e)}{\leq}&\sum_{p=1}^{P_C}H(\mathbf{A}_p)+\sum_{p=1}^{P_C}H(\mathbf{B}_p)-P_C\frac{TS}{m}\log|\mathbb{F}|-P_C\frac{SD}{n}\log|\mathbb{F}|\\
\overset{(f)}{=}&P_C\frac{TS}{m}\log|\mathbb{F}|+P_C\frac{SD}{n}\log|\mathbb{F}|-P_C\frac{TS}{m}\log|\mathbb{F}|\\
&-P_C\frac{SD}{n}\log|\mathbb{F}|\\
=&0,
\end{aligned}
\tag{3.26}
$$

where (a) follows from the definition of encoding functions, since $\mathbf{A}_{\mathcal{P}}$ is a deterministic function of $\mathbf{A}$ and $\mathbf{R}_p$, and $\mathbf{B}_{\mathcal{P}}$ is a deterministic function of $\mathbf{B}$ and $\mathbf{R}'_p$, respectively, for all $p \in [1, P_C]$; (b) follows from Equations (3.23) and (3.24), since from P_R polynomial evaluations $\mathbf{A}_{\mathcal{P}}$ and $\mathbf{B}_{\mathcal{P}}$ in Equations (3.23) and (3.24) we can recover $2P_C$ unknowns when the coefficients $\mathbf{A}_{i,j}$ and $\mathbf{B}_{k,l}$ are known, given that we have $P_R \geq 2P_C$; (c) and (d) follows since $\mathbf{R}_p$ and $\mathbf{R}'_p$ are independent uniformly distributed entries; (e) follows by upper bounding the joint entropy using the sum of individual entropies;

and (f) follows from an argument similar to (d). Hence, the proposed scheme is information-theoretically secure. □

Remark 2. *When $P_C \geq 1$ a direct application of the GPD construction in Figure 3.3 would yield the larger recovery threshold*

$$P_R = \begin{cases} t^* s d^* + s - 1, & \textit{if } \Delta_{P_C} = \frac{P_C}{s}, \\ d s t^* + s - 1 - 2(s\Delta_{P_C} - P_C), & \textit{if } \Delta_{P_C} > \frac{P_C}{s}. \end{cases} \tag{3.27}$$

3.4.2 Secure Generalized PolyDot Code: The $s \geq t$ Case

As illustrated in Figure 3.6, when $s \geq t$, we instead augment input matrices $\mathbf{A}$ and $\mathbf{B}$ by adding

$$\Delta'_{P_C} \triangleq \left\lceil \frac{P_C}{\min\{t, d\}} \right\rceil \tag{3.28}$$

column and row blocks to matrices $\mathbf{A}$ and $\mathbf{B}$. This can be seen to yield a smaller recovery threshold. Accordingly, the $t \times s^*$ augmented block matrix $\mathbf{A}^* = [\mathbf{A}\ \mathbf{R}]$ with $s^* = s + \Delta'_{P_C}$ is obtained as

$$\mathbf{A}^* = \begin{bmatrix} \mathbf{A}_{1,1} & \dots & \mathbf{A}_{1,s} & \mathbf{R}_{1,1} & \dots & \mathbf{R}_{1,\Delta'_{P_C}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{t,1} & \dots & \mathbf{A}_{t,s} & \mathbf{R}_{t,1} & \dots & \mathbf{R}_{t,\Delta'_{P_C}} \end{bmatrix}, \tag{3.29}$$

while the $s^* \times d$ augmented block matrix $\mathbf{B}^*$ is defined as

$$\mathbf{B}^* = \begin{bmatrix} \mathbf{R}' \\ \mathbf{B} \end{bmatrix} = \begin{bmatrix} \mathbf{R}'_{\Delta'_{P_C},1} & \dots & \mathbf{R}'_{\Delta'_{P_C},d} \\ \vdots & \ddots & \vdots \\ \mathbf{R}'_{1,1} & \dots & \mathbf{R}'_{1,d} \\ \mathbf{B}_{1,1} & \dots & \mathbf{B}_{1,d} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{s,1} & \dots & \mathbf{B}_{s,d} \end{bmatrix}. \tag{3.30}$$

As for (3.29) and (3.30), if $\Delta'_{P_C} - P_C/\min\{t,d\} > 0$, the last $s\Delta'_{P_C} - P_C$ block matrices in Equation (3.29), with bottom-to-top right-to-left ordering in $\mathbf{R}$, and in Equation (3.30) with right-to-left top-to-bottom ordering in $\mathbf{R}'$, are all-zero block matrices. The construction of sequences $\mathbf{a}^*$ and $\mathbf{b}^*$ is analogous to the GPD in the non-secure case. In particular, as seen in Figure 3.6, the time block sequence $\mathbf{a}^*$ is

$$\mathbf{a}^* = \{\mathbf{a}_1, \mathbf{r}_1, \mathbf{a}_2, \mathbf{r}_2, \ldots, \mathbf{a}_t, \mathbf{r}_t\}, \tag{3.31}$$

whereas the block sequence $\mathbf{b}^*$ is defined as

$$\mathbf{b}^* = \{\mathbf{b}_1, \mathbf{0}, \mathbf{b}_2, \ldots, \mathbf{0}, \mathbf{b}_d, \hat{\mathbf{0}}, \mathbf{r}'_{\Delta'_{P_C}}, \ldots, \mathbf{r}'_1\}. \tag{3.32}$$

Here, $\mathbf{0}$ and $\hat{\mathbf{0}}$ are a block sequence of t and $t-1$ all-zero block matrices with dimensions $S/s \times D/d$, respectively, while $\mathbf{r}'_i$ is the ith row of the random matrix $\mathbf{R}'$.

Theorem 6. *For a given security level $P_C < P$, the proposed SGPD code achieves the recovery threshold*

$$P_R = t(s^*d - \Delta'_{P_C}) + ts + 2P_C - 1 \tag{3.33}$$

and the communication load in Equation (3.16)*, where $s^* = s + \Delta'_{P_C}$ for any integer values t, s, and d such that $s \geq t$, $m = ts$, and $n = sd$.*

Proof. We define the z-transform of sequences $\mathbf{a}^*$ and $\mathbf{b}^*$ respectively as

$$\begin{aligned}\mathbf{F}_{\mathbf{a}^*}(z) = & \sum_{i=1}^{t}\sum_{j=1}^{s} \mathbf{A}^*_{i,j} z^{i-1+t(j-1)} \\ & + \sum_{i=1}^{t}\sum_{j=s+1}^{s^*} \mathbf{A}^*_{i,j} z^{i-1+t(j-1)},\end{aligned} \tag{3.34}$$

$$\begin{aligned}\mathbf{F}_{\mathbf{b}^*}(z) = & \sum_{k=1+\Delta'_{P_C}}^{s^*}\sum_{l=1}^{d} \mathbf{B}^*_{k,l} z^{(s^*-k)t+ts^*(l-1)} \\ & + \sum_{k=1}^{\Delta'_{P_C}}\sum_{l=1}^{d} \mathbf{B}^*_{k,l} z^{t(s^*d-\Delta'_{P_C})+d(\Delta'_{P_C}-k)+l-1}.\end{aligned} \tag{3.35}$$

The (i,l) block $\mathbf{C}_{i,l} = \sum_{r=1}^{s} \mathbf{A}_{i,r}\mathbf{B}_{r,l}$, for all $i \in [1,t]$ and $l \in [1,d]$, of matrix $\mathbf{C} = \mathbf{AB}$ can be seen equal to the $(i-1+t(s^*l-1))$th sample of the convolution $\mathbf{c}^* = \mathbf{a}^* * \mathbf{b}^*$. The rest of the proof follows in a manner akin to Theorem 5. □

Remark 3. *The computational complexity of SGPD codes for both workers and master server can be summarized as follows. Each worker is assigned to compute the multiplication* $\mathbf{C}_p = \mathbf{A}_p\mathbf{B}_p$, *requiring* $TSD/(tsd)$ *multiplications. For the master server, encoding matrices* $\mathbf{A}_p$ *and* $\mathbf{B}_p$ *at each worker amounts to evaluating* z*-transforms* $\mathbf{F}_{\mathbf{a}^*}(z)$ *and* $\mathbf{F}_{\mathbf{b}^*}(z)$ *at a random point* z_p. *This requires multiplying* z_p *by* $(ts+P_C)$ *and* $(sd+P_C)$ *submatrices, each of dimension* $T/t \times S/s$ *and* $S/s \times D/d$, *respectively. This requires* $P_C(TS/(ts)+SD/(sd))+TS+SD$ *multiplications. Overall, the master server needs to carry out* $PP_C(TS/(ts) + SD/(sd)) + P(TS + SD)$ *multiplications. For decoding, the master server interpolates a polynomial degree* P_R-1 *for each element in* $\mathbf{C}$. *Using a polynomial interpolation algorithm, the decoding complexity amounts to* $(P_R-1)(\log(P_R-1))^2TD/(td)$ *multiplications [64].*

Example 2. *We now provide some numerical results of the proposed SGPD scheme. We set* $P = 3000$ *workers and parameters* $m = n = 36$. *The trade-off between communication load* C_L *and recovery threshold* P_R *for both non-secure conventional GPD codes* ($P_C = 0$) *and proposed SGPD code with colluding workers* $P_C = 11$ *and* $P_C = 29$ *is illustrated in Figure 3.7. The figure quantifies the loss in terms of achievable pairs* (P_R, C_L) *that is caused by the security constraint.*

3.4.3 Trading Off Computation and Communication Latencies

In this subsection, we elaborate on the importance of enabling a flexible trade-off between communication load and recovery threshold by analyzing the overall completion time for the matrix multiplication task at hand. The completion delay is the sum of latencies due to computation and communication.

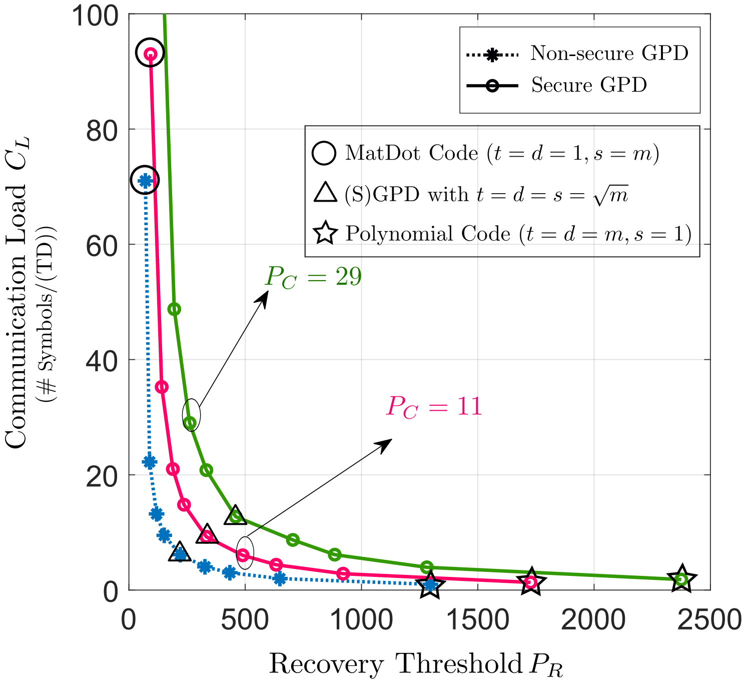


Figure 3.7 Communication load C_L versus recovery threshold P_R for both non-secure generalized PolyDot (GPD) and secure generalized PolyDot (SGPD) codes ($m = n = 36$ and $P = 3000$ workers).

To this end, following a well-established model [66], we assume that computation at each worker p requires a random time T_p^{comp}, measured in some specified unit of time, that is modeled as a shifted exponential distribution with cumulative distribution function (cdf)

$$F^{\text{comp}}(T^{\text{comp}}) = 1 - \exp\left(-\frac{\mu TSD}{tsd}(T^{\text{comp}} - T^{\text{comp}}_{\text{min}})\right), \tag{3.36}$$

for $T \geq T^{\text{comp}}_{\text{min}}$ and $F^{\text{comp}}(T) = 0$ otherwise. According to (3.36), the parameter $T^{\text{comp}}_{\text{min}}$ represents the minimum processing time, and $1/\mu$ represents the average excess computing time, with respect to $T^{\text{comp}}_{\text{min}}$, per multiplication (recall Remark 3). Assuming independent computing times, for a given recovery threshold P_R, the computation time T^{comp} is hence given as the P_Rth-order statistic, i.e., the P_Rth smallest variable, among the i.i.d. variables $(T_1^{\text{comp}}, \ldots, T_P^{\text{comp}})$. Its expectation is

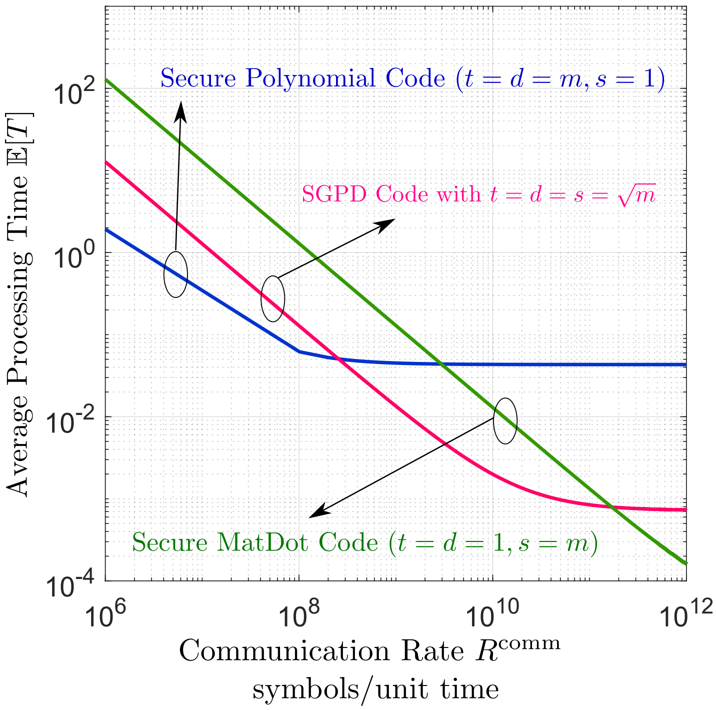


Figure 3.8 Average completion time $\mathbb{E}[T]$ versus communication rate R^{comm} for secure generalized PolyDot (SGPD) codes with $P = 3000$, $P_C = 29$, $T = S = D = 1008$, $\mu = 0.5 \times 10^{-4}$, and $T^{\text{comp}} = 1$, and $m = n = 36$: *(i)* $t = d = 36$, $s = 1$ (SGPD code), *(ii)* $t = s = d = 6$, and *(iii)* $t = d = 1$, $s = 36$ (secure MatDot code).

given by [89]

$$\mathbb{E}[T^{\text{comp}}] = \frac{tsd}{\mu TSD} \sum_{i=1}^{P_R} \frac{1}{P - P_R + i} = \frac{tsd}{\mu TSD}(H_P - H_{P-P_R}), \tag{3.37}$$

where H_P is the generalized harmonic number defined as $H_P = \sum_{i=1}^{P} 1/i$.

Suppose now that the workers communicate with the master server are a link with an overall download rate R^{comm} (symbols per unit time). The communication latency is hence given as

$$T^{\text{comm}} = P_R \frac{TD}{tdR^{\text{comm}}}, \tag{3.38}$$

since the workers need to return $P_R TD/(td)$ symbols to the master server. Overall, the average completion time is given as

$$\mathbb{E}[T] = T_{\min}^{\text{comp}} + \frac{tsd}{\mu TSD}(H_P - H_{P-P_R}) + P_R \frac{TD}{tdR^{\text{comm}}}. \tag{3.39}$$

Example 3. *Let consider* $P = 3000$ *workers and parameters* $m = n = 36$*. We assume that* $P_C = 29$*,* $T = S = D = 1008$*,* $\mu = 0.5 \times 10^{-4}$*, and* $T_{\min}^{comm} = 1$*. We compare the performance of the following SGPD codes:* (i) $t = d = 36$ *and* $s = 1$ *(secure Polynomial code);* (ii) $t = s = d = 6$*;* (iii) $t = d = 1$ *and* $s = 36$ *(secure MatDot code). The values of* C_L *and* P_R *for these codes are shown in Figure 3.7. The average completion time* (3.39) *is plotted versus the communication rate* R^{comm} *in Figure 3.8. The figure shows that the optimal choice of the latency-minimizing SGPD code along the curve in Figure 3.7 depends on the system's operating point: For small communication rates, it is preferable to reduce the communication load* C_L*, and hence secure Polynomial codes are the best choice; while for large communication rate, it is optimal to choose codes with an increasingly large value of the communication load* C_L*.*

3.5 Secure and Private Generalized PolyDot Code

In this section, we study the setup shown in Figure 3.2. We propose a variant of the private and secure GPD code introduced in [61] that we refer to as private and secure GPD (PSGPD) code. Note that in [61] a private coded matrix multiplication scheme is proposed only for Polynomial codes with $s = 1$ in (3.11). We derive the corresponding achievable set of pairs (P_R, C_L) as defined in Section 3.2 under the condition $P_C = 1$, i.e., the workers do not collude.

Theorem 7. *For a given security level $P_C = 1$, there is an achievable PSGPD codes with the recovery threshold*

$$P_R = \begin{cases} s(t+1)d, & \text{if } s < t, \\ ts(d+1) - t + 1, & \text{if } s \geq t, \end{cases} \tag{3.40}$$

and the communication load (3.16)*, for any integer values* t, s, *and* d *such that* $m = ts$, *and* $n = sd$.

Proof. We start by discussing the $s < t$ case, as done in Section 3.4. The polynomial encoding function for the input matrix $\mathbf{A}$, is obtained is defined as in (3.23) for $P_C = 1$, that is

$$\mathbf{F}_{\mathbf{A}}(z) = \sum_{i=1}^{t} \sum_{j=1}^{s} \mathbf{A}_{i,j} z^{s(i-1)+(j-1)} + \mathbf{R} z^{st}, \tag{3.41}$$

where we recall that $\mathbf{R}$ is an $T/t \times S/s$ random matrix with i.i.d. uniform random elements in $\mathbb{F}$. The encoded matrices are given as $\mathbf{A}_p^{(\kappa)} = \mathbf{F}_{\mathbf{A}}(z_{\kappa,p})$ for values $z_{\kappa,p}$ to be discussed below. For the desired index κ, the master server also computes the query vector $\mathbf{q}_p^{(\kappa)}$ for all $p \in [1, P]$. This is obtained as

$$\mathbf{q}_p^{(\kappa)} = [z_1, \ldots, z_{\kappa-1}, z_{\kappa,p}, z_{\kappa+1}, \ldots, z_L], \tag{3.42}$$

where all points $\{z_i\}_{i \neq \kappa}$ are selected uniformly i.i.d. from $\mathbb{F}$ but are identical for all p. The points $\{z_{\kappa,p}\}_{p=1}^{P}$ are selected i.i.d. as distinct elements from $\mathbb{F}$ (recall that we have $|\mathbb{F}| > P$). We note that, as in the private information retrieval scheme [61], the query vector (3.42) does not leak any information on index κ in the sense defined by condition (3.9). The master server evaluates $\mathbf{F}_{\mathbf{A}}(z)$ in (3.41) at the distinct random point $z_{\kappa,p}$, to produce the encoded matrices $\mathbf{A}_p^{(\kappa)} = \mathbf{F}_{\mathbf{A}}(z_{\kappa,p})$, and then sends $\mathbf{A}_p^{(\kappa)}$ along with the query vector $\mathbf{q}_p^{(\kappa)}$ to worker $p \in [1, P]$.

Each worker p, after receiving the query vectors $\mathbf{q}_p^{(\kappa)}$, encodes the library $\mathcal{B}$ into a matrix $\mathbf{B}_p^{(\kappa)}$ as follows. Define the polynomial encoding function for each matrix

$\mathbf{B}^{(r)}$, $r \in [1, L]$, in the library $\mathcal{B}$ as in (3.24) for $P_C = 0$, i.e.,

$$\mathbf{F}_{\mathbf{B}^{(r)}}(z) = \sum_{k=1}^{s} \sum_{l=1}^{d} \mathbf{B}_{k,l}^{(r)} z^{s-k+(l-1)s(t+1)}. \tag{3.43}$$

Each worker p computes the encoded matrices as

$$\begin{aligned} \mathbf{B}_p^{(\kappa)} \triangleq & \sum_{r \in [1,L]} \mathbf{F}_{\mathbf{B}^{(r)}}([\mathbf{q}_p^{(\kappa)}]_r) \\ & = \mathbf{F}_{\mathbf{B}^{(\kappa)}}(z_{\kappa,p}) + \sum_{r \in [1,L] \backslash \kappa} \mathbf{F}_{\mathbf{B}^{(r)}}(z_r), \end{aligned} \tag{3.44}$$

where $[\mathbf{q}_p^{(\kappa)}]_r$ denotes the rth element of the query vector $\mathbf{q}_p^{(\kappa)}$.

After encoding the library, each worker p computes the matrix product $\mathbf{C}_p^{(\kappa)} = \mathbf{A}_p^{(\kappa)} \mathbf{B}_p^{(\kappa)}$ and then sends $\mathbf{C}_p^{(\kappa)}$ back to the master server. We note that both polynomials $\mathbf{F}_{\mathbf{A}}(z)$ and $\mathbf{F}_{\mathbf{B}^{(\kappa)}}(z)$, assigned to the input matrix $\mathbf{A}$ and the desired matrix $\mathbf{B}^{(\kappa)}$, are evaluated at the same random points $z_{\kappa,1}, \ldots, z_{\kappa,P}$ for workers $1, \ldots, P$, respectively. Since each undesired matrix is evaluated at an identical random point for all workers the second term in (3.44), i.e., $\sum_{r \in [1,L] \backslash \kappa} \mathbf{F}_{\mathbf{B}^{(r)}}(z_r)$, can be considered as a constant term.

To reconstruct all blocks $\mathbf{C}_{i,l}^{(\kappa)}$ of the product matrix $\mathbf{C}^{(\kappa)} = \mathbf{A}\mathbf{B}^{(\kappa)}$, the master server carries out polynomial interpolation, upon receiving a number of multiplication results equal to at least $\deg(\mathbf{F}_{\mathbf{A}}(z)\mathbf{G}_{\mathbf{B}^{(\kappa)}}(z)) + 1$, which is $s(t+1)d$, for the case $s < t$.

Similarly, for the $s \geq t$ case, the polynomial encoding function for the input matrix $\mathbf{A}$ as in (3.34) for $P_C = 1$, that is,

$$\mathbf{F}_{\mathbf{A}}(z) = \sum_{i=1}^{t} \sum_{j=1}^{s} \mathbf{A}_{i,j} z^{i-1+t(j-1)} + \mathbf{R} z^{ts}, \tag{3.45}$$

and the encoding function for matrices $\mathbf{B}^{(r)}$ is given as in (3.35) for $P_C = 0$, that is

$$\mathbf{F}_{\mathbf{B}^{(r)}}(z) = \sum_{k=1}^{s} \sum_{l=1}^{d} \mathbf{B}_{k,l}^{(r)} z^{(s-k)t+ts(l-1)}. \tag{3.46}$$

The encoded matrices $\mathbf{A}_p^{(\kappa)}$ and $\mathbf{B}_p^{(\kappa)}$ are defined as above, and so are the query vectors $\mathbf{q}_p^{(\kappa)}$ for all $p \in [1, P]$.

The security of the data matrix $\mathbf{A}$ against non-colluding workers is guaranteed by appending the random matrix $\mathbf{R}$ to the input matrix $\mathbf{A}$ in (3.41) in the same way as described in Section 3.4. The details for both cases $s < t$ and $s \geq t$ are given in the proofs of Theorems 5 and 6, respectively, for the case of $P_C = 1$. The privacy condition of (3.9) follows by definition of the query vectors (3.42) for the desired index $\kappa \in [1, L]$, as proved in [61]. Finally, the recovery threshold and the communication load follow in a manner analogous to Theorems 5 and 6. □

Remark 4. *The computational complexity of PSGPD codes for both workers and master server is summarized as follows. In PSGPD codes, each worker has two duties, namely encoding the library $\mathcal{B}$ and computing the multiplication $\mathbf{C}_p^{(\kappa)} = \mathbf{A}_p^{(\kappa)}\mathbf{B}_p^{(\kappa)}$. Encoding the library, i.e., computing the matrix $\mathbf{B}_p^{(\kappa)}$ in* (3.44)*, requires to evaluate $\mathbf{F}_{\mathbf{B}^{(r)}}(z)$, $r \in [1, L]$ at query vector $\mathbf{q}_p^{(\kappa)}$. Hence, the former task requires LSD multiplications, while the latter entails $TSD/(tsd)$ multiplications. In total, each worker carries out $LSD + TSD/(tsd)$ multiplications. The master server encodes matrix $\mathbf{A}_p^{(\kappa)}$ with $(1 + ts)TS/(ts)$ multiplications. In total, for all P workers, the master server needs $P(1+ts)TS/(ts)$ multiplications. The computation complexity of the decoding complexity of the master server is the same as for SGPD codes, namely $\mathcal{O}((P_R - 1)(\log(P_r - 1))^2 TD/(td)))$.*

Example 4. *Let us consider $P = 3000$ workers and parameters $m = n = 36$. We assume that $P_C = 1$ in order to compare the performance of proposed SGPD and PSGPD codes. Note that both recovery threshold and communication load of the PSGPD code do not depend on the number of public matrices $|\mathcal{B}| = L$ in the library. The trade-off between communication load C_L and recovery threshold P_R is illustrated in Figure 3.9 for both codes. The figure shows that, for a fixed value of P_R, the resulting achievable value of the communication load C_L is smaller for PSGPD than*

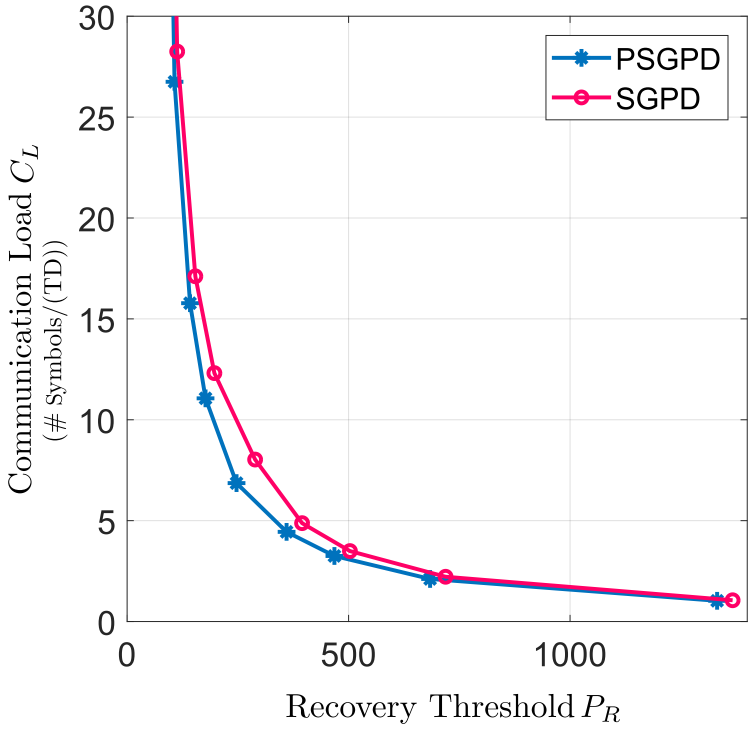


Figure 3.9 Communication load C_L versus recovery threshold P_R for secure generalized PolyDot (SGPD) codes with $P_C = 1$ and private and secure generalized PolyDot (PSGPD) codes ($m = n = 36$ and $P = 3000$ workers).

for SGPD codes. This suggests that the privacy requirement on the index κ imposed by PSGPD is less demanding than the security constraint on matrix $\mathbf{B}$ *under which SGPD codes operate.*

Remark 5. *As for SGPD codes, the overall average completion time of PSGPD codes can be derived following the same steps as described in Section 3.4.3.*

3.6 Discussion and Concluding Remarks

In this chapter, we have considered the problem of secure and private distributed matrix multiplication on $\mathbf{C} = \mathbf{AB}$ in terms of design of computational codes for two settings. In the first setting, the two matrices $\mathbf{A}$ and $\mathbf{B}$ contain confidential data and must be kept secure from the workers; and in the second setting , matrix $\mathbf{A}$ is confidential, while matrix **B** is selected in a private manner from a library of public matrices. For both problems, this work presents the best currently known trade-off between communication load and recovery threshold. This is done by presenting two code constructions that generalize the state-of-the-art GPD codes [39, 37, 35], in combination with private information retrieval based codes [61].

CHAPTER 4

CONCLUDING REMARKS AND FUTURE DIRECTIONS

In this dissertation, the problem of using codes to speed up distributed computing systems is studied.

In Chapter 2, we study distributed computing framework, when the input files distributedly stored on the uplink of a cloud radio access network architecture. It focuses in which decoding at the cloud takes place via network function virtualization on commercial off-the-shelf servers. In order to mitigate the impact of straggling decoders in this platform, a novel coding strategy is proposed, whereby the cloud re-encodes the received frames via a linear code before distributing them to the decoding processors. Transmission of a single frame is considered first, and upper bounds on the resulting frame unavailability probability as a function of the decoding latency are derived by assuming a binary symmetric channel for uplink communications. Then, the analysis is extended to account for random frame arrival times. In this case, the trade-off between an average decoding latency and the frame error rate is studied for two different queuing policies, whereby the servers carry out per-frame decoding or continuous decoding, respectively. Numerical examples demonstrate that the bounds are useful tools for code design and that coding is instrumental in obtaining a desirable compromise between decoding latency and reliability.

In Chapter 3, we consider large matrix multiplications. These operations are often carried out on a distributed computing platform with a master server and multiple workers in the cloud operating in parallel. For such distributed platforms, in addition to exact recovery requirements, security and privacy constraints on the data matrices are imposed, and the recovery threshold as a function of the communication load is studied. First, it is assumed that both matrices contain private information and that workers can collude to eavesdrop on the content of these data matrices. For this problem, a novel class of secure codes is introduced, referred to as secure generalized PolyDot codes, that generalize state-of-the-art non-secure codes for matrix multiplication. Secure generalized PolyDot codes allow a flexible trade-off between recovery threshold and communication load for a fixed maximum number of colluding workers while providing perfect secrecy for the two data matrices. Then, a connection between secure matrix multiplication and private information retrieval is studied. It is assumed that one of the data matrices is taken from a public set known to all the workers. In this setup, the identity of the matrix of interest should be kept private from the workers. For this model, a variant of generalized PolyDot codes is presented that can guarantee both secrecy of one matrix and privacy for the identity of the other matrix for the case of no colluding servers.

In summary we show that

- Coded distributed computing systems allows reliable and timely channel decoding in a C-RAN architecture based on distributed unreliable processors rather than uncoded ones;
- Coding can provide a systematic way to add redundancy into distributed algorithms so that their runtime is not affected by stragglers;
- We introduce the dependency graph of a linear code and its chromatic number as novel relevant parameters of a linear code and;
- We introduce a novel class of secure codes, referred to as secure generalized PolyDot codes. We also propose a secure and private class of codes called private ans secure generalized PolyDot codes for matrix multiplication.

4.1 Future Research Directions

Among interesting open problems, we mention the design of optimal NFV codes and the extension of the principle of NFV coding to other channels. Note that the approach proposed here applies directly to other additive noise channels in which the user code is an additive group. A key example is the additive Gaussian channel with lattice codes at the user.

Also, we focus on private secure generalized PolyDot schemes for any number of colluding workers that provides a smaller computational complexity at the workers. Finally, the establishment of a converse bound and the consideration of nonperfect communication channels between workers and master server are open problems.

REFERENCES

[1] European Telecommunications Standards Institute. Network function virtualisation (NFV); report on models and features for end-to-end reliability. Technical Report GS NFV-REL 003, Apr., 2016.

[2] European Telecommunications Standards Institute. Cloud RAN and MEC: A perfect pairing. ISBN No. 979-10-92620-17-7, Feb., 2018.

[3] Martin Abadi, Joan Feigenbaum, and Joe Kilian. On hiding information from an oracle. *Journal of computer and system sciences*, 39(1):21–50, Aug., 1989.

[4] Mehmet Fatih Aktas, Pei Peng, and Emina Soljanin. Effective straggler mitigation: Which clones should attack and when? *ACM SIGMETRICS Performance Evaluation Review*, 45(2):12–14, Sep., 2017.

[5] Ali Al-Shuwaili, Osvaldo Simeone, Joerg Kliewer, and Petar Popovski. Coded network function virtualization: Fault tolerance via in-network coding. *IEEE Wireless Communications Letters*, 5(6):644–647, Dec., 2016.

[6] Malihe Aliasgari, Jörg Kliewer, and Osvaldo Simeone. Coded computation against processing delays for virtualized cloud-based channel decoding. *IEEE Transaction on Communication*, 67(1):28–38, Jan., 2019.

[7] Malihe Aliasgari, Jörg Kliewer, and Osvaldo Simeone. Coded computation against straggling decoders for network function virtualization. In *Proceeding IEEE International Symposium on Information Theory (ISIT)*, pages 711–715, Jun., 2018.

[8] Malihe Aliasgari, Osvaldo Simeone, and Jörg Kliewer. Distributed and private coded matrix computation with flexible communication load. In *Proceeding IEEE International Symposium Information Theory (ISIT)*, pages 1092–1096, Jul., 2019.

[9] Islam Alyafawi, Eryk Schiller, Torsten Braun, Desislava Dimitrova, Andre Gomes, and Navid Nikaein. Critical issues of centralized and cloudified LTE-FDD radio access networks. In *IEEE International Conference on Communications (ICC)*, pages 5523–5528. IEEE, Jun., 2015.

[10] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings Federation of Information Processing Societies*, pages 483–485, Apr., 1967.

[11] Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, and Ion Stoica. Effective straggler mitigation: Attack of the clones. In *Proceeding of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, volume 13, pages 185–198, Apr., 2013.

[12] Ganesh Ananthanarayanan, Srikanth Kandula, Albert G Greenberg, Ion Stoica, Yi Lu, Bikas Saha, and Edward Harris. Reining in the outliers in Map-Reduce clusters using mantri. In *Proceeding of the 10th USENIX Symposium on Operating Systems Design and Implementation*, volume 10, page 24, Oct., 2010.

[13] Navid Azizan-Ruhi, Farshad Lahouti, Amir Salman Avestimehr, and Babak Hassibi. Distributed solution of large-scale linear systems via accelerated projection-based consensus. *IEEE Transactions on Signal Processing*, 67(14):3806–3817, Dec., 2019.

[14] Karim Banawan and Sennur Ulukus. The capacity of private information retrieval from coded databases. *IEEE Transaction on Information Theory*, 64(3):1945–1956, Mar., 2018.

[15] Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Locally random reductions: Improvements and applications. *Journal of Cryptology*, 10(1):17–36, Sep., 1997.

[16] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Ilan Orlov. Share conversion and private information retrieval. In *IEEE 27th Conference on Computational Complexity*, pages 258–268, Jun., 2012.

[17] Yitzhak Birk and Tomer Kol. Coding on demand by an informed source (iscod) for efficient broadcast of different supplemental data to caching clients. *IEEE Transactions on Information Theory*, 52(6):2825–2830, Jan., 2006.

[18] George Blakley. Safeguarding cryptographic keys. In *International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318. IEEE, Jun., 1979.

[19] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013.

[20] Rowland Leonard Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(02):194–197, Jul., 1941.

[21] Lynn Elliot Cannon. *A cellular computer to implement the Kalman filter algorithm.* PhD thesis, Montana State University-Bozeman, College of Engineering, Aug., 1969.

[22] Wei-Ting Chang and Ravi Tandon. On the capacity of secure distributed matrix multiplication. *arXiv preprint, arXiv:1806.00469*, 2018.

[23] Wei-Ting Chang and Ravi Tandon. On the upload versus download cost for secure and private matrix multiplication. *arXiv preprint, arXiv:1906.10684*, 2019.

[24] Manmohan Chaubey and Erik Saule. Replicated data placement for uncertain scheduling. In *IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 464–472. IEEE, May., 2015.

[25] Jaeyoung Choi, David W Walker, and Jack J Dongarra. Pumma: Parallel universal matrix multiplication algorithms on distributed memory concurrent computers. *Concurrency: Practice and Experience*, 6(7):543–570, Oct., 1994.

[26] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50, Oct., 1995.

[27] Anindya B Das, Aditya Ramamoorthy, and Namrata Vaswani. Random convolutional coding for robust and straggler resilient distributed matrix computation. *arXiv preprint, arXiv:1907.08064*, 2019.

[28] Jeffrey Dean and Luiz André Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, Feb., 2013.

[29] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communication of the ACM*, 51(1):107–113, Feb., 2008.

[30] Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, Dec., 1974.

[31] Alexandros G Dimakis, Kannan Ramchandran, Yunnan Wu, and Changho Suh. A survey on network codes for distributed storage. *Proceedings of the IEEE*, 99(3):476–489, 2011.

[32] Rafael GL D'Oliveira, Salim El Rouayheb, and David Karpuk. GASP codes for secure distributed matrix multiplication. *arXiv preprint, arXiv:1812.09962*, 2018.

[33] Jack Dongarra, Thomas Herault, and Yves Robert. Fault tolerance techniques for high-performance computing. In *Computer Communications and Networks*, pages 3–85. Springer, Jul., 2015.

[34] Uwe Dötsch, Mark Doll, Hans-Peter Mayer, Frank Schaich, Jonathan Segel, and Philippe Sehier. Quantitative analysis of split base station processing and determination of advantageous architectures for LTE. *Bell Labs Technical Journal*, 18(1):105–128, Dec., 2013.

[35] Sanghamitra Dutta, Ziqian Bai, Haewon Jeong, Tze Meng Low, and Pulkit Grover. A unified coded deep neural network training strategy based on generalized polydot codes for matrix multiplication. *arXiv preprint, arXiv:1811.10751*, 2018.

[36] Sanghamitra Dutta, Viveck Cadambe, and Pulkit Grover. Short-dot: Computing large linear transforms distributedly using coded short dot products. In *Advances In Neural Information Processing Systems*, pages 2100–2108, Dec., 2016.

[37] Sanghamitra Dutta, Mohammad Fahim, Farzin Haddadpour, Haewon Jeong, Viveck Cadambe, and Pulkit Grover. On the optimal recovery threshold of coded matrix multiplication. *arXiv preprint, arXiv:1801.10292*, 2018.

[38] Mohammad Fahim and Viveck R Cadambe. Numerically stable polynomially coded computing. *arXiv preprint, arXiv:1903.08326*, 2019.

[39] Mohammad Fahim, Haewon Jeong, Farzin Haddadpour, Sanghamitra Dutta, Viveck Cadambe, and Pulkit Grover. On the optimal recovery threshold of coded matrix multiplication. In *Proceeding 55th Allerton Conference on Communication, Control, and Computing, IL, USA*, pages 1264–1270, Oct., 2017.

[40] Joan Feigenbaum. Encrypting problem instances. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 477–488. Springer, Aug., 1985.

[41] R Freij-Hollanti, O. W. Gnilke, C Hollanti, and D. A. Karpuk. Private information retrieval from coded databases with colluding servers. *SIAM Journal on Applied Algebra and Geometry*, 1(1):647–664, Nov., 2017.

[42] Kristen Gardner, Samuel Zbarsky, Sherwin Doroudi, Mor Harchol-Balter, and Esa Hyytia. Reducing latency via redundant requests: Exact analysis. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):347–360, Jun., 2015.

[43] William Gasarch. A survey on private information retrieval. *Bulletin of the EATCS*, 82(113):72–107, Feb., 2004.

[44] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and System Sciences*, 60(3):592–629, Jun., 2000.

[45] Thomas Herault and Yves Robert. *Fault-tolerance techniques for high-performance computing.* Springer, Jul., 2015.

[46] Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in NFV: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, Mar., 2016.

[47] Wassily Hoeffding. A class of statistics with asymptotically normal distribution. In *Breakthroughs in Statistics*, pages 308–334. Springer, 1992.

[48] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[49] Kuang-Hua Huang and Jacob A Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Transaction on Computers*, 100(6):518–528, Jun., 1984.

[50] Svante Janson. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24(3):234–248, Mar., 2004.

[51] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint, arXiv:1506.08473*, 2015.

[52] Zhuqing Jia and Syed A Jafar. Cross subspace alignment codes for coded distributed batch matrix multiplication. *arXiv preprint, arXiv:1909.13873*, 2019.

[53] Zhuqing Jia and Syed A Jafar. On the capacity of secure distributed matrix multiplication. *arXiv preprint, arXiv:1908.06957*, 2019.

[54] Gauri Joshi, Yanpei Liu, and Emina Soljanin. On the delay-storage trade-off in content download from coded distributed storage systems. *IEEE Journal on Selected Areas in Communications*, 32(5):989–997, Dec., 2014.

[55] Gauri Joshi, Emina Soljanin, and Gregory Wornell. Efficient redundancy techniques for latency reduction in cloud systems. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 2(2):12, Sep., 2017.

[56] Jaber Kakar, Seyedhamed Ebadifar, and Aydin Sezgin. On the capacity and straggler-robustness of distributed secure matrix multiplication. *IEEE Access*, 7:45783–45799, Apr., 2019.

[57] Jinkyu Kang, Osvaldo Simeone, and Joonhyuk Kang. On the trade-off between computational load and reliability for network function virtualization. *IEEE Communications Letters*, 21:1767–1770, Dec., 2017.

[58] Fatemeh Kazemi, Esmaeil Karimi, Anoosheh Heidarzadeh, and Alex Sprintson. Private information retrieval with private coded side information: The multi-server case. *arXiv preprint, arXiv:1906.11278*, 2019.

[59] Fatemeh Kazemi, Esmaeil Karimi, Anoosheh Heidarzadeh, and Alex Sprintson. Single-server single-message online private information retrieval with side information. In *Proceeding IEEE International Symposium on Information Theory (ISIT)*, pages 350–354, Jul., 2019.

[60] Shahrouz Khalili and Osvaldo Simeone. Uplink HARQ for cloud RAN via separation of control and data planes. *IEEE Transactions on Vehicular Technology*, 66(5):4005–4016, Mar., 2017.

[61] Minchul Kim and Jungwoo Lee. Private secure coded computation. *arXiv preprint, arXiv:1902.00167*, 2019.

[62] Ger Koole and Rhonda Righter. Resource allocation in grid computing. *Journal of Scheduling*, 11(3):163–173, Aug., 2008.

[63] Jack Kosaian, KV Rashmi, and Shivaram Venkataraman. Learning a code: Machine learning for approximate non-linear coded computation. *arXiv preprint, arXiv:1806.01259*, 2018.

[64] Hsiang-Tsung Kung. *Fast evaluation and interpolation.* Carnegie Mellon University, Tech. Rep., 2009.

[65] Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran. Speeding up distributed machine learning using codes. *Proceeding IEEE International Symposium on Information Theory*, pages 1143–1147, Jul., 2016.

[66] Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran. Speeding up distributed machine learning using codes. *IEEE Transation on Information Theory*, 64(3):1514–1529, Mar., 2018.

[67] Kangwook Lee, Ramtin Pedarsani, and Kannan Ramchandran. On scheduling redundant requests with cancellation overheads. *IEEE/ACM Transactions on Networking*, 25(2):1279–1290, Apr., 2017.

[68] Kangwook Lee, Changho Suh, and Kannan Ramchandran. High-dimensional coded matrix multiplication. In *Proceeding IEEE International Symposium Information Theory (ISIT)*, pages 2418–2422, Jun., 2017.

[69] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *Proceeding of the 11th USENIX Conference on Operating Systems Design and Implementation, OSDI*, volume 14, pages 583–598, Oct., 2014.

[70] Songze Li, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. A unified coding framework for distributed computing with straggling servers. In *Globecom Workshops (GC Wkshps), 2016 IEEE*, pages 1–6. IEEE, Dec., 2016.

[71] Songze Li, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. Coded MapReduce. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 964–971. IEEE, Oct., 2015.

[72] Songze Li, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. Coded distributed computing: Straggling servers and multistage dataflows. In *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*, pages 164–171. IEEE, Oct., 2016.

[73] Songze Li, Mohammad Ali Maddah-Ali, Qian Yu, and A Salman Avestimehr. A fundamental tradeoff between computation and communication in distributed

computing. *IEEE Transactions on Information Theory*, 64(1):109–128, May., 2018.

[74] Songze Li, Mohammad Ali Maddah-Ali, Qian Yu, and A Salman Avestimehr. A fundamental tradeoff between computation and communication in distributed computing. *IEEE Transaction on Information Theory*, 64(1):109–128, Sep., 2017.

[75] Jiajia Liu, Zhongyuan Jiang, Nei Kato, Osamu Akashi, and Atsushi Takahara. Reliability evaluation for NFV deployment of future mobile broadband networks. *IEEE Wireless Communications*, 23(3):90–96, Apr., 2016.

[76] Ankur Mallick, Malhar Chaudhari, and Gauri Joshi. Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication. *arXiv preprint, arXiv:1804.10331*, 2018.

[77] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, Dec., 2016.

[78] Navid Nikaein. Processing radio access network functions in the cloud: Critical issues and modeling. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services,*, pages 36–43. ACM, Apr., 2015.

[79] Navid Nikaein, Raymond Knopp, Florian Kaltenberger, Lionel Gauthier, Christian Bonnet, Dominique Nussbaum, and Riadh Ghaddab. OpenAirInterface: an open LTE network in a PC. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 305–308. ACM, Sep., 2014.

[80] Hanzaleh Akbari Nodehi and Mohammad Ali Maddah-Ali. Secure coded multi-party computation for massive matrix operations. *arXiv preprint, arXiv:1908.04255*, 2019.

[81] Hanzaleh Akbari Nodehi and Mohammad Ali Maddah-Ali. Limited-sharing multi-party computation for massive matrix operations. In *Proceeding IEEE International Symposium on Information Theory (ISIT)*, pages 1231–1235, Jun., 2018.

[82] Linus Nyman and Mikael Laakso. Notes on the history of fork and join. *IEEE Annals of the History of Computing*, 38(3):84–87, 2016.

[83] Yury Polyanskiy, H Vincent Poor, and Sergio Verdú. Channel coding rate in the finite blocklength regime. *IEEE Transactions on Information Theory*, 56(5):2307–2359, Dec., 2010.

[84] Amirhossein Reisizadehmobarakeh, Saurav Prakash, Ramtin Pedarsani, and Salman Avestimehr. Coded computation over heterogeneous clusters. [Online] www.arxiv.org, arXiv:1701.05973 [cs.IT], 2017.

[85] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, Oct., 2010.

[86] David P Rodgers. Improvements in multiprocessor system design. *ACM SIGARCH Computer Architecture News*, 13(3):225–231, Jun., 1985.

[87] Veronica Quintuna Rodriguez and Fabrice Guillemin. Towards the deployment of a fully centralized cloud-RAN architecture. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*, pages 1055–1060, Valencia, Spain, Jun., 2017.

[88] Veronica Quintuna Rodriguez and Fabrice Guillemin. Cloud-ran modeling based on parallel processing. *IEEE Journal on Selected Areas in Communications*, 36(3):457–468, Nov., 2018.

[89] Sheldon M Ross. *Introduction to Probability Models*. Academic Press, 2014.

[90] Peter Rost and Athul Prasad. Opportunistic hybrid arqenabler of centralized-RAN over nonideal backhaul. *IEEE Wireless Communications Letters*, 3(5):481–484, Dec., 2014.

[91] Abdón Sánchez-Arroyo. Determining the total colouring number is NP-hard. *Discrete Mathematics*, 78(3):315–319, 1989.

[92] Albin Severinson, Alexandre Graell i Amat, and Eirik Rosnes. Block-diagonal coding for distributed computing with straggling servers. In *Information Theory Workshop (ITW)*, pages 464–468, Nov., 2017.

[93] Nihar B Shah, Kangwook Lee, and Kannan Ramchandran. When do redundant requests reduce latency? *IEEE Transactions on Communications*, 64(2):715–722, Sep., 2015.

[94] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, Nov., 1979.

[95] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, Jul., 1948.

[96] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10, May., 2010.

[97] Edgar Solomonik and James Demmel. Communication-optimal parallel 2.5 d matrix multiplication and lu factorization algorithms. In *European Conference on Parallel Processing*, pages 90–109. Springer, Aug., 2011.

[98] Adarsh M Subramaniam, Anoosheh Heidarzadeh, and Krishna R Narayanan. Random khatri-rao-product codes for numerically-stable distributed matrix multiplication. *arXiv preprint, arXiv:1907.05965*, 2019.

[99] Hua Sun and Syed Ali Jafar. The capacity of private information retrieval. *IEEE Transaction on Information Theory*, 63(7):4075–4088, Jul., 2017.

[100] Behrooz Tahmasebi and Mohammad Ali Maddah-Ali. Private sequential function computation. *arXiv preprint, arXiv:1908.01204*, 2019.

[101] Rashish Tandon, Qi Lei, Alexandros Dimakis, and Nikos Karampatziakis. Gradient coding: Avoiding stragglers in synchronous gradient descent. [Online] www.arxiv.org arXiv:1612.03301 [cs.IT], 2016.

[102] Rashish Tandon, Qi Lei, Alexandros G Dimakis, and Nikos Karampatziakis. Gradient coding: Avoiding stragglers in distributed learning. In *International Conference on Machine Learning*, pages 3368–3376, Aug., 2017.

[103] Henk C Tijms. *A First Course in Stochastic Models*. John Wiley and Sons, Jul., 2003.

[104] Robert A Van De Geijn and Jerrell Watts. Summa: Scalable universal matrix multiplication algorithm. *Concurrency: Practice and Experience*, 9(4):255–274, Oct., 1997.

[105] Richard Walker. Implementing discrete mathematics: combinatorics and graph theory with mathematica. *The Mathematical Gazette*, 76(476):286–288, Jul., 1992.

[106] Da Wang, Gauri Joshi, and Gregory Wornell. Using straggler replication to reduce latency in large-scale parallel computing. *ACM SIGMETRICS Performance Evaluation Review*, 43(3):7–11, Jun., 2015.

[107] Heecheol Yang and Jungwoo Lee. Secure distributed computing with straggling servers using polynomial codes. *IEEE Transaction on Information Forensics and Security*, 14(1):141–150, Jan., 2019.

[108] Yaoqing Yang, Malhar Chaudhari, Pulkit Grover, and Soummya Kar. Coded iterative computing using substitute decoding. *arXiv preprint, arXiv:1805.06046*, 2018.

[109] Yaoqing Yang, Pulkit Grover, and Soummya Kar. Computing linear transformations with unreliable components. *IEEE Transactions on Information Theory*, 63(6):3729–3756, Mar., 2017.

[110] Sergey Yekhanin. Private information retrieval. *Commun. ACM*, 53(4):68–73, Apr., 2010.

[111] Qian Yu, Mohammad Maddah-Ali, and Salman Avestimehr. Polynomial codes: An optimal design for high-dimensional coded matrix multiplication. In *Advances in Neural Information Processing Systems*, pages 4403–4413, Dec., 2017.

[112] Qian Yu, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding. *arXiv preprint, arXiv:1801.07487*, 2018.

[113] Qian Yu, Netanel Raviv, Jinhyun So, and A Salman Avestimehr. Lagrange coded computing: Optimal design for resiliency, security and privacy. *arXiv preprint, arXiv:1806.00939*, 2018.

[114] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *Proceeding of the 2nd USENIX Conference on Hot topics in Cloud Computing*, pages 10–10, Jun., 2010.