CODES FOR PRIVATE DISTRIBUTED COMPUTATION WITH APPLICATIONS TO MACHINE LEARNING

by

RAWAD BITAR

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Salim El Rouayheb

And approved by

New Brunswick, New Jersey

January, 2020

ABSTRACT OF THE DISSERTATION

Codes for Private Distributed Computation with Applications to Machine Learning

By RAWAD BITAR

Dissertation Director:

Salim El Rouayheb

We consider the problem of private distributed computation. Our main interest in this problem stems from machine learning applications. A master node (referred to as master) possesses a tremendous amount of *confidential* data (e.g., personal, genomic or medical data) and wants to perform intensive computations on it. The master divides these computations into smaller computational tasks and distribute them to *untrusted* workers that perform these tasks in parallel. The workers return their results to the master, who processes them to obtain its original task. In large scale systems, the appearance of slow and unresponsive workers, called stragglers, is inevitable. Stragglers incur large delays on the computation if they are not accounted for. Our goal is to construct codes that maintain the privacy of the data and allow flexible straggler mitigation, i.e., tolerate the presence of a varying number of stragglers.

We propose the use of communication efficient secret sharing (CE-SS) in private *linear* coded computation with straggler mitigation. A CE-SS scheme satisfies the properties of threshold secret sharing. Moreover, it allows the master to reconstruct the secret by reading and communicating the minimum amount of information from a variable number of workers, up to a given threshold. We introduce three explicit constructions of CE-SS codes called *Staircase codes*. The first construction achieves optimal communication and read costs for a given number of workers d. The second construction achieves optimal costs universally for all possible values of d between k and n. The third construction, which is the most general, achieves optimal costs universally for all values of d in any given set $\Delta \subseteq \{k, \dots, n\}$.

We analyze the performance of Staircase codes in distributed computation systems where the workers have fixed resources. We model the workers service time by *i.i.d.* shifted exponential random variables. We derive upper and lower bounds on the Master's mean waiting time. We derive the distribution of the Master's waiting time, and its mean, for systems with up to two stragglers. We show that Staircase codes always outperform existing solutions based on classical secret sharing codes. We validate our results with extensive implementation on Amazon EC2.

We consider the case where the workers have time-varying resources. We develop a private and rateless adaptive coded computation (PRAC) algorithm for distributed matrix-vector multiplication. PRAC is based on the use of Fountain codes coupled with MDS codes to maintain privacy and to adaptively assign tasks to the workers. The assigned tasks are proportional to the estimated resources at the workers. We provide theoretical guarantees on the performance of PRAC and compare it to baselines. Moreover, we validate our theoretical results through simulations and implementation on Android devices.

We go beyond linear coded computation and tackle the problem of distributed gradient descent for general convex loss functions in the presence of stragglers. However, we drop the privacy constraints on the master's data. We propose an approximate gradient coding scheme called *Stochastic Gradient Coding* (SGC), which works when the stragglers are random. SGC distributes data points redundantly to workers according to a pair-wise balanced design, and then simply ignores the stragglers. We prove that the convergence rate of SGC mirrors that of batched Stochastic Gradient Descent (SGD) for the ℓ_2 loss function, and show how the convergence rate can improve with the redundancy. We also provide bounds for more general convex loss functions. We show empirically that SGC requires a small amount of redundancy to handle a large number of stragglers and that it can outperform existing approximate gradient codes when the number of stragglers is large.

We study private information retrieval (PIR) which is an extension of private distributed computation. The data is public and stored (replicated) at the workers. The master wants to retrieve a file without revealing its identity to the workers. Robust PIR capacity is the minimum amount of information downloaded by the master to retrieve the file in the presence of a fixed number of stragglers. We show that communication efficient secret sharing achieve robust PIR capacity simultaneously for all number of stragglers, up to a given threshold. As a result, we construct a family of universally robust PIR called Staircase-PIR.

Acknowledgements

Glory to God in the highest, on earth peace and joy to people. First, and foremost I thank God for this wondrous journey full of passion, success, and failure. Devotedly learning and attempting to turn failures into success would have been a lot harder without God's ubiquitous help.

I am indebted to my family. The footsteps of my parents and two brothers are the cornerstone of my personality, and their continuous support is the foundation of my achievements. I am also forever grateful to Maroun Hayek and Hayate Hayek for their parental love and for providing a comforting home in New York.

It is a pleasure to have Dr. Salim El Rouayheb as my Ph.D. advisor. He made this long and tough journey flow smoothly. Even though I was his first student, he was equipped and ready to help overcome all the hurdles of my Ph.D. Dr. El Rouayheb, thanks for all those long meetings full of brainstorming and fruitful discussions. You are an inspiration. You taught me from scratch how to formulate practical problems and how to form an intuition about each solution. You revamped my presentation skills and kept trying to pass your reader-friendly writing style, despite this task being hard for students with my background.

A special thanks to all my collaborators, you seasoned my thesis. Prof. Sidharth Jaggi, thank you for the various pointers on tackling research problems. You enlarged my problem-solving arsenal. Dr. Mary Wootters quickly formulating your intuition is impressive. Thanks for showing me how to connect coding theory to convergence analysis, a building block of my research. Dr. Parimal Parag, you redefined the concept of being available. Thank you for meeting at night to compensate for the time difference. Dr. Hulya Seferoglu, thank you for explaining how to tackle complexities arising from implementing coding theoretical tools in networks; Collaborating with your group widened my horizon. Dr. Venkat Dasari, thank you for introducing application-oriented research problems. Moreover, thank you and the Army Research Lab for the generous funding. Dr. Yasaman Keshtkarjahromi, thank you for producing exceptional ideas. Mr. Yuxuan Xing, thank you for implementing PRAC on Android devices. I never thought I would see my research ideas implemented on a smartphone. Mr. Peiwen Tian, thank you for implementing Staircase codes on Amazon EC2. Mr. Serge Kas Hanna, we may have written one paper together, yet our daily discussions in the lab greatly influenced my research.

I am lucky to have a considerate and reliable defense committee. Despite attending my talks on several occasions, Prof. Emina Soljanin, Dr. Predrag Spasojevic, and Prof. Roy Yates responded quickly to my emails and accepted to be on my comprehensive exam committee, Ph.D. proposal committee, and Ph.D. defense committee. Thank you for engaging with my overlapping talks and providing constructive feedback. Prof. Antonia Wachter-Zeh, my gratitude for e-attending the defense from a different continent goes beyond my ability to express it in words. I also thank Prof. Anand Sarwate, who was so kind to be a member of my comprehensive exam committee and Ph.D. proposal committee.

Traveling around the world and living in different countries seasoned my journey; Thanks to everyone who made it possible. In particular, I would like to thank Prof. Sidharth Jaggi for inviting me to spend a semester at the Chinese University of Hong Kong. Living and doing research in the far-east is a marvelous experience. Moreover, I would like to thank Dr. El Rouayheb for extending the invitations he received. In an equal manner, I would like to thank Prof. Camilla Hollanti and Prof. Rafael Schaefer for generously accommodating me at their respective institutions and for providing a welcoming research environment. Thank you, Prof. Marcelo Firer, for inviting me to participate in the Saõ Paulo coding school. I learned a lot and made many connections within the information theory community.

Influential characters shaped the path leading to my Ph.D. In particular, I thank Mr. Charbel Hayek, Mr. Simon Farsah, Dr. Georges Salloum, Dr. Nabil Karkour and Dr. Gilles Ballouz for molding my career. Your guidance is invaluable.

I started my Ph.D. at the Illinois Institute of Technology (IIT) and transferred to Rutgers University alongside my advisor. I thank Ms. Darnita Murphy and Ms. Bonnie Dow for being always happy to help. Darnita, thank you for going out of your way to provide solutions in a matter of hours, if not minutes! Further, I thank Dr. Dong Jin, Prof. Guillermo Atkin, and Prof. Yongyi Yang for forming a wonderful Ph.D. proposal committee at IIT and for providing valuable feedback improving my thesis. Furthermore, thanks to Ms. Christy Lafferty, Ms. Pamela Heinolds, Ms. Tea Akins, and Mr. John Scaffidi for producing a smooth transfer and a welcoming atmosphere at Rutgers. Christy, thank you for meticulously answering my complicated questions! John, thanks for printing all those posters!

I am blessed to have supportive, enthusiastic, and lighthearted friends by my side. They are the propeller that keeps pushing forward irrespective of the surrounding conditions. Through their actions, they sustained my eager to fulfill this chapter of my life. Sabine El Hajj, because of our daily chats, I started to regain my optimism. Thank you for your continuous encouragement and for helping me overcome my weaknesses. Serge Kas Hanna, it is a pleasure to be your friend, lab-mate, flat-mate, and gym partner for four consecutive years. You helped in every aspect of this journey. The time we spent was as enjoyable as the first Kung Pao chicken we had in Chicago. Georges Hourane, Anthony Antoun, Ralph Salem, and Elie Harfoush, you guys are simply amazing. Together, we made the best memories! Gus Sleiman and Shatha Abbassi, your friendship is indispensable; You are the beating heart of New Jersey. Razane Tajeddine, it feels like we started our journeys yesterday. The lab is not the same without junk dinners from 7-Eleven and Lebanese TV shows. Xiao Huang, thank you for being a magnificent, selfless, and supportive friend. Linda Feldmann, you are the best landlord. Thank you for hosting me and treating me like family throughout the three years I spent in Chicago. Sreechakra Goparaju, you were never hesitant to offer guidance, even though I have known you briefly.

I enjoyed helping with the book chapter; you convert stressful work into fun times. Rafael D'Oliveira, Ghadir Ayache, Carolina Naim, and Fangwei Ye, you vivify the walls surrounding our desks. Fangwei, thank you for your guidance during my stay in HK. Peiwen Tian, you are an exceptional colleague and friend; Next time we compete in a Basketball tournament, we should win. Ashutosh Jain and Abhay Kulkarni, we were destined to meet in IIT, become friends, and move to the same town in New Jersey. Thank you for the fun road trips and activities. Shrihari Sankarasubramanian, thanks for the tips to survive the windy city.

Dedication

To my family Tony, Colette, Jad and Imad.

Table of Contents

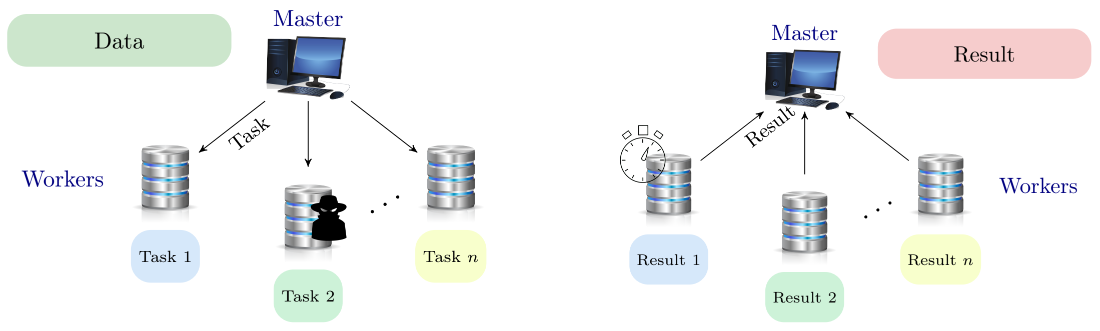
List of Figures

List of Tables

Chapter 1

Introduction

In the era of big data, most people's activities are migrating to the digital world. Social media, online consumer services (shopping, making reservations, etc), online financial services (banking, investing, etc), wearable devices and internet of things are mere examples of how much people rely on information and communications technology. A tremendous amount of information is being generated daily [2]. Running machine learning algorithms on this data is computationally intensive. To speed up the computation, the master node possessing the data (referred to as



(a) Master sends the tasks to the workers. (b) Workers send the results to the master.

Figure 1.1: Distributed computing system with n worker machines. The master wants to run large computation on its data. In a first step the master divides the large computation task into n smaller tasks and sends them to the workers to compute those tasks in parallel. In a second step the workers return their results to the master who combines them to obtain the computation of its original task. Challenges of distributed computing systems are privacy and straggler mitigation. *Privacy*: any subset of workers does not learn anything about the data from the tasks assigned to them. *Straggler mitigation*: the master tolerates the presence of a slow or unresponsive workers.

master) divides the computational task into smaller tasks and distribute them to *untrusted* worker machines. The workers perform the smaller computations in parallel and then return their results to the master, who can process them to obtain the result of its original task.

A distributed computing system is depicted in Figure 1.1. In this thesis, we focus on the following challenges of distributed computing, from an information theoretic point of view.

Information theoretic data privacy: Distributing computations to untrusted workers raises privacy and reliability concerns [3,4]. In many applications, the master wants to run computations on sensitive data that must be kept private from the workers, such as data generated from wearable devices, genomic data, and medical data. We require information theoretic privacy, i.e., we impose no constraints on the computational power of the compromised workers. The assumption that we have to make here is a limit on the number of workers colluding against the master.

Straggler mitigation: The overall computing time at the master is affected by the variability in the computation times of the workers. Accounting for the slowest workers referred to as stragglers significantly reduces the overall computation time [5].

Computation setup Consider the following optimization problem. Let A be an $m \times \ell$ matrix representing the data of the master. Let $\mathbf{y}$ be a vector of length m representing the labels of the data, i.e., the j^{th} row vector $\mathbf{a}_j$ of A is labeled by the j^{th} coordinate y_j of $\mathbf{y}$. The master wants to find a vector $\mathbf{x}^\star$ that minimizes a given loss function $\mathcal{L}(A, \mathbf{x}, \mathbf{y})$, i.e.,

$$\mathbf{x}^\star = \min_{\mathbf{x} \in \mathbb{F}_q^\ell} \mathcal{L}(A, \mathbf{x}, \mathbf{y}).$$

This minimization is solved using the iterative gradient descent algorithm. The master starts with a random guess $\mathbf{x}_0$ and updates it at iteration $t \geq 1$ as follows

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t \nabla \mathcal{L}(A, \mathbf{x}_t, \mathbf{y}). \tag{1.1}$$

Here, $t \geq 1$, $\nabla \mathcal{L}(A, \mathbf{x}_t, \mathbf{y})$ is the gradient of the loss function and γ_t is a parameter of the algorithm. Consider the ℓ_2 loss function where $\mathcal{L}(A, \mathbf{x}, \mathbf{y}) = 1/2 \|A\mathbf{x} - \mathbf{y}\|_2^2$, which is the loss used in many algorithms, such as principal component analysis, support vector machines, and other gradient-descent based algorithms [6, 7]. In this case equation (1.1) becomes

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t A^T (A\mathbf{x} - \mathbf{y}). \tag{1.2}$$

The superscript T denotes the transpose of a matrix. This update rule consists of two matrix-vector multiplications, namely $A\mathbf{x}$ and $A^T(A\mathbf{x} - \mathbf{y})$. Throughout the thesis (except for Chapter 5) we focus on linear computations, namely we focus on the multiplication of A by the vector $\mathbf{x}$, and the remaining follows. We focus on maintaining the privacy of the matrix A and not the vector $\mathbf{x}$.

Our goals We look at the private coded computing problem from a coding theoretic perspective. We want to construct codes that allow the master to run private linear computation and: *(i)* maintain the privacy of A; *(ii)* tolerate stragglers; and *(iii)* minimize the overhead spent on ensuring privacy. Beyond matrix-vector multiplication, we study the effect of codes on the number of iterations in the distributed gradient descent setting.

Related works We briefly explain the works that are closely related to this thesis and highlight the ideas of interest.

Secret sharing [8, 9] can be readily used to construct codes for matrix-vector multiplication that achieve information theoretic privacy and mitigate the effect of a given number of stragglers as we illustrate in the next example.

Example 1.1. *Assume the master has $n = 3$ workers available to help compute $A\mathbf{x}$. The master wants to ensure privacy of A against individual workers and wants to tolerate at most $n - k = 1$ straggler. The following 3 shares $(R, A + R, A + 2R)$ form a secret sharing scheme, with R being an $m \times \ell$ random matrix, called key, chosen uniformly at random and independently of A. The* *master sends one share to each worker, let* $S_1 = R$*,* $S_2 = A + R$ *and* $S_3 = A + 2R$*. The master sends* $\mathbf{x}$ *to each worker, the workers compute* $S_i\mathbf{x}$ *and send the result to the master. The master can decode* $A\mathbf{x}$ *after receiving any two responses. For instance, if worker* 3 *is a straggler the master obtains* $A\mathbf{x} = S_2\mathbf{x} - S_1\mathbf{x}$*. Privacy is ensured, because the matrix* A *is padded with the key in each share.*

In classical secret sharing the master ignores the computations of $n - k$ workers and has to download and decode the introduced randomness used for privacy. In Example 1.1, even if there were no stragglers, the master ignores one of the computations and decodes both $R\mathbf{x}$ and $A\mathbf{x}$ from any two full responses of the workers. Hence, more delays are incurred by spending communication and computation resources on decoding $R\mathbf{x}$, which is only needed for privacy.

We want to construct codes that allow the master to: *(i)* maintain the privacy of A; *(ii)* tolerate a flexible amount of stragglers; *(iii)* use partial computations from the workers to reduce delays; and *(iv)* minimize the resources spent on computing and communicating the randomness used for privacy. We extend our ideas to private function retrieval where the data is public and owned by the workers and the master wants to compute a private function of this data.

Gradient coding [10] are coding theoretic schemes targeted for straggler mitigation in distributed gradient descent applications [11–13]. The goal is to allow the master to tolerate a fixed number of stragglers and still decode the gradient of the loss function. The number of tolerated stragglers depends on the amount of computation assigned to the workers (redundancy). Approximate gradient coding allows the master to tolerate a variable number of stragglers for a given redundancy at the expense of computing an approximation of the gradient [14–17]. Approximating the gradient increases the number of iterations required by the master to reach the desired vector $\mathbf{x}^\star$.

We want to design an approximate gradient coding scheme that *(i)* requires small redundancy; *(ii)* allows the master to wait for the fastest few workers to compute an estimate of the gradient; and *(iii)* enjoy better convergence rate than the previously proposed schemes.

Organization The rest of this Chapter is organized as follows. In Section 1.1, we give an overview of the thesis and summarize the main results (also shown in Table 1.1). In Sections 1.2 to 1.6, we introduce in more details the problems studied in each chapter and explain their connection to our goals. Along the way, we outline our main contributions. The chapters are meant to be self-contained; the reader may skip to the chapter of preference and need not read the thesis sequentially. In Section 1.7, we list the publications that resulted from this thesis.

1.1 Overview of the Thesis

The thesis is divided into the following chapters:

- *Chapter 2:* We focus on communication efficient secret sharing problem. The master wants to share a secret with n workers such that a legitimate user contacting the workers to decode the secret can tolerate any number of stragglers, up to a given threshold. In addition, for every number of stragglers, the workers read and communicate the minimum overhead of randomness to the user who decodes only this overhead and the stored secret. We construct Staircase codes, a new family of communication efficient secret sharing.
- *Chapter 3:* We analyze the performance of Staircase codes in private distributed computation. We assume that the workers have similar resources. We model the computation time

Chapter	Topic	Contribution
Chapter 2	Communication efficient secret sharing	Construct Staircase codes that allow flexible straggler mitigation and minimize the communication cost.
Chapter 3	Private coded computing via Staircase codes	Analyze the performance of Staircase codes in private coded computing where the workers have similar resources and show they outperform existing schemes .
Chapter 4	Dynamic private coded computing	Construct PRAC, a private and adaptive rateless code that allow the master to adapt to the different and time-varying resources of the workers.
Chapter 5	Approximate gradient coding	Introduce *Stochastic Gradient Coding* (SGC) which enjoys good convergence speed, high flexibility in the number of stragglers and small redundancy.
Chapter 6	Private information retrieval	Construct Staircase-PIR codes that simultaneously achieve the minimum download cost for any number of stragglers.

Table 1.1: Overview of the results of this thesis.

of the workers by *i.i.d.* shifted exponential random variables. We are the first to analyze the performance of private coded computation with straggler mitigation. We show that Staircase codes outperform methods based on classical secret sharing schemes.

- *Chapter 4:* We consider the case where the resources at the workers are different and change dynamically. We introduce PRAC, a private rateless and adaptive coding technique that adapts to the time-varying resources at the workers. The idea is to use Fountain codes that are rateless and couple them with MDS codes to achieve privacy. We show that, under this model, PRAC outperform previously introduced private coded computation schemes.

- *Chapter 5:* We consider the approximate gradient coding problem with no privacy constraints on the data. We introduce a new family of codes that we call stochastic gradient codes (SGC). We show that SGC requires a small redundancy, allows a high flexibility in straggler mitigation and enjoys a good convergence rate that mirrors the convergence rate of centralized stochastic gradient descent.

- *Chapter 6:* We study private information retrieval (PIR) with flexible straggler mitigation. The main cost of PIR is the amount of information downloaded by the master as function of the number of stragglers. We show that communication efficient secret sharing achieve the minimum download cost simultaneously for all number of stragglers, up to a given threshold. As a result, we construct Staircase-PIR, a family of PIR codes that achieve minimum download cost simultaneously for all number of stragglers.

Remark 1.1. Throughout this thesis we focus on maintaining the privacy of the data A. In some applications, $\mathbf{x}$ may contain information about A and need to be protected as well. We explain how all our schemes can be generalized to such settings in Appendix A. In fact, the problem of securing both A and $\mathbf{x}$ simultaneously is an active research topic, see for example [18–21].

1.2 Communication Efficient Secret Sharing

We study this problem in Chapter 2. Consider the private matrix-vector multiplication problem. Information theoretic privacy of the master's data and straggler mitigation can be insured by encoding A using classical secret sharing [8, 9]. Secret sharing allows the master to tolerate the presence of up to $n-k$ stragglers for a fixed parameter $k<n$ and guarantees the privacy of the data against any $z<k$ colluding workers. We illustrate the idea in Example 1.1.

We want to construct secret sharing schemes that provide the master flexibility in the number of stragglers, up to a given threshold. Besides, we know that secret sharing schemes designed for fewer stragglers requires less randomness because the amount of randomness used to encode the data is proportional to $k-z$. For fixed z, increasing k, i.e., tolerating less $n-k$ stragglers, allows the reduction of randomness. Hence, we want secret sharing schemes with parameters n, k and z that allow the master to decode the result from any d workers, $k \leq d \leq n$, while downloading the amount of randomness required by a secret sharing scheme with parameter n, $k=d$ and z.

Such secret sharing schemes are known as communication efficient sharing schemes (CE-SS). The existence of CE-SS and a lower bound on the amount of randomness needed to decode the desired result are shown in [22, 23]. CE-SS schemes that allow the master to tolerate 0 or $n-k$ stragglers are given in [23]. The idea in constructing such schemes is to divide the shares given to the workers into small sub shares as illustrated in Example 1.2.

Contribution We construct universal Staircase codes [24, 25] for all parameters n, k and z such that $z<k\leq n$. The master contacts any d workers, $k \leq d \leq n$ to decode the secret while achieving the information theoretic minimum communication cost. In other words, the master can tolerate any number of stragglers up to $n-k$. Privacy is guaranteed against any z colluding workers. We construct two other families of Staircase codes. The first one allows the master to decode the secret from any k or any d, for a given $d>k$, workers while achieving minimum communication cost. The second one called Δ-universal Staircase codes allows the master to

Worker 1	Worker 2	Worker 3
$A_1 + A_2 + R_1$	$A_1 + 2A_2 + 4r_1$	$A_1 + 3A_2 + 4R_1$
$R_1 + R_2$	$R_1 + 2R_2$	$R_1 + 3R_2$

Table 1.2: An example of a CE-SS code based on the Staircase code construction over $\mathbb{F}_5$ for $n = 3$ workers, threshold $k = 2$ and $z = 1$ colluding workers. The master contacting any $k = 2$ workers downloads all their shares, i.e., 4 matrices in total, in order to decode A. However, if there are no stragglers, the master downloads the first matrix (in blue) of each share, i.e., 3 matrices in total, in order to decode A. Notice that if there are no stragglers, the master can only decode R_1, whereas if there is a straggler the master has to decode R_1 and R_2.

decode the secret from any d workers while achieving minimum communication cost, for any value of $d \in \Delta$, where $\Delta \subseteq \{k, \dots, n\}$.

Example 1.2. *Consider again the secret sharing problem of Example 1.1 with $n = 3$, $k = 2$, $z = 1$. We construct a communication efficient secret sharing based on Staircase codes introduced in Chapter 2. We divide the matrix A into 2 row blocks A_1, A_2 and assume A is uniformly distributed over $\mathbb{F}_5^{m \times \ell}$. We use two keys R_1, R_2 each drawn independently and uniformly at random from $\mathbb{F}_5^{m/2 \times \ell}$. The shares encoded using Staircase codes and given to the workers are shown in Table 1.2.*

The CE-SS scheme enjoys the following properties. First, the master decodes the secret either by contacting any $k = 2$ workers and downloading all their shares, i.e., 4 matrices or by contacting all 3 workers and downloading the first matrix (in blue) of each share, i.e., 3 matrices in total. The key idea here is that the master is only interested in decoding A and not necessarily the keys. When there are no stragglers, $d = 3$, the master decodes A and only the key R_1, whereas when $d = k = 2$, the master has to decode A and both of the keys. The amount of information downloaded by the master achieves the minimum communication cost. Second, privacy is achieved because the matrix A is padded by random keys R_1, R_2 and each $z = 1$ party cannot obtain any information about A_1 and A_2.

In the following, we analyze the performance of private coded computing schemes based on Staircase codes in terms of the delays experienced by the master.

1.3 Private Coded Computing via Staircase Codes

We study this problem in Chapter 3. We want to analyze the performance of private coded computing schemes based on universal Staircase codes. Note that communication is a major bottleneck in many distributed computing systems, such as cloud computing and crowdsourcing. Therefore, minimizing the communication cost at the master plays an important role in reducing the aggregate delays.

We focus on the setting where the workers have similar computation power and a similar network connection. We assume that the service times of the workers are independent and identically distributed (*i.i.d.*) random variables following a shifted exponential distribution. This assumption is per the literature on coded computing [6, 26]. Staircase codes divide the computation tasks assigned to the workers into smaller sub-tasks so the workers have more flexibility in sending partial results to the master. The master keeps receiving partial results from the workers until it can decode its original computation. Hence, an additional assumption we impose is that the service time of a worker (time to compute and send one task to the master) is equally divided between the sub-tasks.

Contribution We analyze the waiting time at the master under *i.i.d.* shifted exponential distribution of the workers' service time [27,28]. We derive bounds on the mean waiting time at the master and show that universal Staircase codes always outperform classical secret sharing. We validate our results with extensive simulations on MATLAB and extensive implementations on Amazon EC2 clusters.

As a consequence of our results, universal Staircase codes outperform any private distributed computing scheme that imposes a threshold on the number of stragglers. To the extent of our knowledge, the work in [27] is the first to analyze latency for private distributed coded computing under the presence of stragglers.

We illustrate the scheme based on Staircase codes in the following example.

Example 1.3. *Consider the same setting of Example 1.2 with $n = 3$, $k = 2$ and $z = 1$. We* *divide the matrix A into 2 row blocks A_1, A_2 and assume A is uniformly distributed over $\mathbb{F}_5^{m\times\ell}$. We use two keys R_1, R_2 each drawn independently and uniformly at random from $\mathbb{F}_5^{m/2\times\ell}$. The shares encoded using Staircase codes introduced in Chapter 2 and given to the workers are shown in Table 1.2.*

The master sends $\mathbf{x}$ to the workers. Each worker multiplies $\mathbf{x}$ by its sub-shares sequentially from top to bottom and sends the result to the master independently. The master has two possibilities to decode $A\mathbf{x}$: 1) the master receives the multiplication of the first sub-share (in blue) from all the workers; or 2) the master receives the multiplication of the full shares from any two workers. The master sends a stop message to the workers after receiving enough multiplication results to decode $A\mathbf{x}$. The master here has flexibility in the number of stragglers and achieves the minimum communication cost for any number of stragglers. The overall waiting time at the master is the minimum between the time all workers send back the result of the first subtask to the master and the time any two workers send the results of all their subtasks to the master.

The main assumption of this setting that may nod hold in other scenarios is the assumption of *i.i.d.*service times of the workers. We tackle the case where workers have different service times in the next section.

1.4 Adaptive Private Coded Computation

We study this problem in Chapter 4. Here we consider private coded computation on workers that have different and time-varying resources. Resources of the workers depend on many factors such as the location of the workers, network congestion, computation workload and worker battery life. Applications include clusters with variability in the computation workload and network congestion at the workers. Other applications include Internet of Things networks and Edge computing where the workers also enjoy great variability due to the difference in the nature of the devices (phone, tablet, sensor, etc).

In this setting, the key assumption of *i.i.d.* service times at the workers does not hold. Thus, schemes pre-allocating equal tasks to the workers do not fully utilize the computation resources at the workers. We want to design coding schemes that adapt to the variability of the resources and reduce the waiting time at the master. The idea is to use Fountain codes [29–31] coupled with an MDS code to ensure privacy. Fountain codes are a family of rateless codes that allow the master to keep generating and sending tasks to the workers until it receives enough answers to decode $A\mathbf{x}$.

Previous works did not consider the privacy constraint on the master's data. In [32], the authors consider a static setting with different resources known a priori to the master. The master assigns tasks to the workers that are proportional to the computation power of each worker. Our codes are based on the work of [33] in which the authors use Fountain codes to dynamically allocate sub-tasks to the workers. The frequency of allocating the sub-tasks to the workers is inversely proportional to their respective estimated service time. This method is shown to adapt to the dynamic behavior of the system and avoid idling times at the workers.

Contribution We construct PRAC, a private and adaptive rateless code designed to allow the master to adaptively allocate tasks to the workers while maintaining the privacy of the data. The code is rateless in the sense that the master can keep generating sub-tasks until it can decode the desired result. We analyze the waiting time of the master when using PRAC. We show that PRAC outperform any private codes that pre-allocate the tasks to the workers, unless the workers have *i.i.d.* service times [34,35]. We supplement our theoretical findings with extensive simulations on MATLAB and implementation on Android devices.

We illustrate PRAC in the following example.

Example 1.4. *We consider the same setting as in Example 1.3, where the master sends tasks to $n = 3$ workers. The master divides A into 3 row blocks A_1, A_2 and A_3; and encodes these matrices using a Fountain code*[1] *[29–31]. Notice here that the master has flexibility in deciding*

[1]Fountain codes are desirable here for two properties: (i) they provide a fluid abstraction of the coded packets so the master can always decode with high probability as long as it collects enough packets; (ii) They have low decoding complexity.

Time	Worker 1	Worker 2	Worker 3
1	R_1	$\mathbf{A_1} + R_1$	$\mathbf{A_3} + R_1$
2		R_2	
3	$\mathbf{A_2} + \mathbf{A_3} + R_2$		
4			

Table 1.3: Example of PRAC in heterogeneous and time-varying setup for $n = 3$ workers. The master divides A into 3 row blocks A_1, A_2 and A_3 and encodes them using Fountain codes to adaptively send tasks to the workers. An instance of Fountain coded codewords is A_1, A_3, $A_2 + A_3$ and $A_1 + A_3$. At first instance, the master generates 2 codewords A_1 and A_3 and generate a random matrix R_1 and send a first subtask to all workers. Notice that each subtask must be "masked" with a new random key to guarantee privacy. When a worker finishes its computation, the master generates a new codeword and possibly a new key and send a new task to that worker. The master keeps generating tasks until it receives enough computations to decode $A\mathbf{x}$. This is possible since the generation of codeword is based on Fountain codes, that are rateless.

on how to divide A, and the threshold k is not needed. An example set of codewords (referred to as packets) is A_1, A_3, $A_2 + A_3$ and $A_1 + A_3$. Before sending a packet to a worker, the master generates a random key matrix R with the same dimensions as A_i with entries drawn uniformly at random from the same alphabet as the entries of A. The key matrix is added to the coded packets to provide privacy as shown in Table 1.3. Now the master adaptively sends the sub-tasks to the workers. In particular, at time slot 1 a key matrix R_1 is created, combined with A_1 and A_3, and transmitted to workers 2 and 3, respectively. R_1 is also transmitted to worker 1 to obtain $R_1\mathbf{x}$ that will help the master in the decoding process. The computation of $(A_1 + R_1)\mathbf{x}$ is completed at the end of time slot 1. Thus, at that time slot, the master generates a new matrix, R_2, and sends it to worker 2. At the end of time slot 2, worker 1 finishes its computation, therefore the master adds R_2 to $A_2 + A_3$ and sends it to worker 1. Now the master waits for worker 2 to return $R_2\mathbf{x}$ and for worker 1 to return $(A_2 + A_3 + R_2)\mathbf{x}$ to decode $A\mathbf{x}$. Note that if worker 3 finishes its computation before worker 1, the master sends $A_1 + A_2 + R_2$ to worker 3 to multiply it with $\mathbf{x}$. Thanks to using key matrices R_1 and R_2, and assuming that workers do not collude, privacy is guaranteed. On a high level, privacy is guaranteed because the observation of the workers is statistically independent of the data A.

So far we considered linear distributed gradient descent, i.e., matrix-vector multiplication. Our ultimate goal is to design private coded computing schemes for general functions needed in machine learning. We start by focusing on non-private distributed stochastic gradient descent.

1.5 Stochastic Gradient Coding for Straggler Mitigation in Distributed Learning

We study this problem in Chapter 5. We consider the setting where the master wants to run a stochastic gradient-descent based algorithm using general convex loss functions. We drop the privacy constraint on the data A. Recall that the master chooses a vector $\mathbf{x}_0$ at random and at iteration $t > 0$ it updates $\mathbf{x}$ as follows

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t \nabla \mathcal{L}(A, \mathbf{x}_t, \mathbf{y}).$$

Here, γ_t is a parameter of the algorithm and $\nabla \mathcal{L}(A, \mathbf{x}_t, \mathbf{y})$ is the gradient of the loss function and is additively separable, i.e.,

$$\nabla \mathcal{L}(A, \mathbf{x}, \mathbf{y}) = \sum_{j=1}^{m} \nabla \mathcal{L}(\mathbf{a}_j, \mathbf{x}, y_j).$$

In this setting, to distribute the computation to the workers, the master sends some of the rows of A to each of the workers without pre-processing. Each worker evaluates the gradient of the loss function on the rows it possesses and sends a linear combination of those evaluations to the master. The master collects the results of the workers and computes $\nabla \mathcal{L}(A, \mathbf{x}_t, \mathbf{y})$.

In the previous sections we assumed that the master needs to compute the exact value of $\nabla \mathcal{L}(A, \mathbf{x}_t, \mathbf{y})$ at each iteration t. However, it is known from the literature on stochastic gradient descent (SGD) [36–38] that the master can compute an estimate of the gradient at iteration t. The estimate is the evaluation of the gradient of the loss function on a small subset of rows $\mathbf{a}_i$ of A. SGD enjoys a faster compute time per iteration as compared to gradient descent (GD), however, it requires more iterations to reach $\mathbf{x}^\star$.

Hence, stragglers mitigation can be done in three ways:

1. Ignore stragglers: The master partitions the rows of A to the workers and simply ignores the stragglers at each iteration. The master computes an estimate of the gradient. This is similar to the work in [39].

2. Gradient coding, e.g., [6,10,12,13]: The master assigns the rows of A with high redundancy to the workers. More precisely, if the master wants to tolerate s stragglers, then each row of A is assigned to $s+1$ workers. The master waits for all but s workers to compute the exact value of the gradient.

3. Approximate gradient coding, e.g., [13–17, 40, 41]: The master assigns the rows of A with low redundancy, independent from the number of stragglers, to the workers. The master waits for all non-stragglers and computes an estimate of the gradient. If less than s workers are stragglers, the master can compute the exact value of the gradient with high probability. If more than s workers are stragglers, the master computes an estimate of the gradient.

All those frameworks minimize the communication cost as each worker sends one vector (linear combination of the evaluations) to the master. However, each framework has its advantages and drawbacks in terms of redundancy (affects computation time per iteration), the total number of iterations spent to reach $\mathbf{x}^\star$ and flexibility in the number of stragglers. On a high-level, the master trades redundancy for the number of iterations spent to reach $\mathbf{x}^\star$ (referred to as convergence). Adding redundancy increases the time spent per iteration, yet decreases the total number of iterations spent to reach $\mathbf{x}^\star$. Gradient coding schemes have good convergence but require high redundancy and do not offer flexibility in the number of stragglers. We focus on approximate gradient coding because it allows the master high flexibility in the number of stragglers and requires a small amount of redundancy. Our motivation for requiring flexibility in the number of stragglers stems from the variability of service time at the workers. Furthermore, we view the straggler phenomena here as the master waiting for a fixed amount of time per iteration. All workers that take more time than decided by the master are considered as stragglers. Therefore, we assume that each worker is a straggler with probability p independently from other workers and independently across iterations.

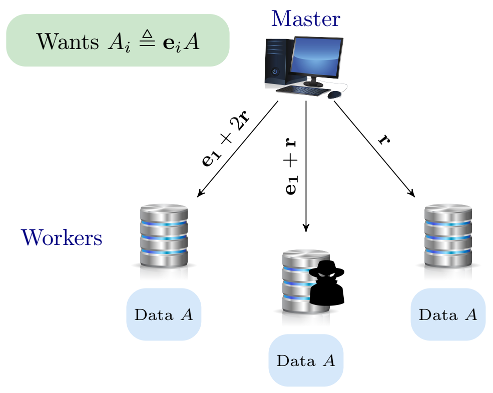
Contribution We introduce an approximate gradient coding scheme called *Stochastic Gradient Coding* (SGC) which works in the random straggler model and which enjoys good convergence speed, high flexibility in the number of stragglers and small redundancy. The estimate of the gradient in SGC is a weighted linear combination of the evaluation of the gradient of the loss function at different rows of A. We analyze the convergence rate of SGC, and we present experimental work which demonstrates that SGC outperforms the most recently proposed schemes [14, 17] when p (the fraction of workers that the master will ignore in each iteration) is relatively large. We show that SGC mirrors results from the literature on SGD, and how redundancy affects the convergence rate of SGC.

Next, we derail from coded computing for machine learning and consider private information retrieval which is an extension of private coded computing.

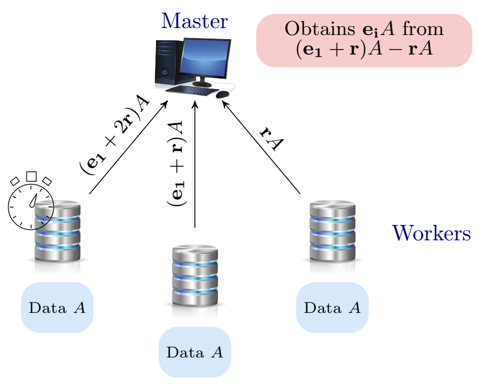
1.6 Universally Robust Private Information Retrieval

We study this problem in Chapter 6. We consider private function retrieval [42] as an extension to private coded computing. The data is public and possessed by n workers and the master wants to compute a private function of this data. A particular case of interest is when the master wants to retrieve a chunk of this public data. This can be thought of as projecting the data on a vector of 0's with a 1 in the position of the chunk of interest. The privacy requirement translates to hiding the identity of the chunk (referred to as file) of interest from any subset of z workers. This problem is known in the literature as private information retrieval (PIR) [43, 44]. The main requirement of PIR is to minimize the amount of information downloaded by the master.

PIR in the presence of stragglers, termed as robust PIR, is studied in the literature, e.g., [45–


Wants $A_i \triangleq \mathbf{e}_i A$
Master
$\mathbf{e}_1 + 2\mathbf{r}$
$\mathbf{e}_1 + \mathbf{r}$
$\mathbf{r}$
Workers
Data A
Data A
Data A

(a) Master sends the queries to the workers.


Master
Obtains $\mathbf{e}_1 A$ from $(\mathbf{e}_1 + \mathbf{r})A - \mathbf{r}A$
$(\mathbf{e}_1 + 2\mathbf{r})A$
$(\mathbf{e}_1 + \mathbf{r})A$
$\mathbf{r}A$
Workers
Data A
Data A
Data A

(b) Workers project the data on the query vector and sends the results to the master.

Figure 1.2: Example of robust private information retrieval with $n = 3$ workers, no collusion and one straggler. The workers own the database A. The master wants to retrieve the ith file $A_i \triangleq A\mathbf{e}_i$ from the database where $\mathbf{e}_i$ is the all-zero vector with 1 in the i^{th} position. The identity of the file must remain hidden from the workers. The master generates a random vector $\mathbf{r}$ and sends the query vectors $\mathbf{r}$, $\mathbf{e}_i + \mathbf{r}$ and $\mathbf{e}_i + 2\mathbf{r}$ to the workers. The workers project the database A on the query vectors and return their results to the master. The master tolerates one straggler and obtains $A\mathbf{e}_i$ from any two responses. For example if the first worker is straggler, the master obtains $A_i = (\mathbf{e}_i + \mathbf{r})A - \mathbf{r}A$ from the remaining two workers. The download cost of this example is equal to 2, because the master downloads two vectors in order to decode A_i. The value of $\mathbf{e}_i$ is hidden from the workers because it is padded by a random vector.

52] and the minimum download cost termed as capacity of robust PIR schemes is characterized in [46,47,51] when the data is replicated among the workers. The common focus of the literature has been on designing robust PIR that are not necessarily universal, which are tailored to a specific number of stragglers $n - k$. We illustrate the idea in Figure 1.2. In [49] the authors present a universally robust PIR scheme for the no collusion case, i.e., $z = 1$, and when the data is stored on the servers using a maximum distance separable (MDS) code.

Contribution We consider the case where the data is replicated among the workers. We bridge communication efficient secret sharing to the problem of robust PIR. We show that all communication efficient secret sharing schemes achieve capacity of robust PIR simultaneously for all number of stragglers. In particular, we construct Staircase-PIR, a family of universally robust PIR that simultaneously achieve the minimum download cost for all number of stragglers, up to a given threshold.

1.7 Publications

We list the publications which results form the main contributions of this dissertation.

J1. Rawad Bitar and Salim El Rouayheb, "Staircase Codes for Secret Sharing with Optimal Communication and Read Overheads," *IEEE Transactions on Information Theory*, Vol. 64, No. 2, February 2018.

C1. Rawad Bitar, Mary Wootters and Salim El Rouayheb, "Stochastic Gradient Coding for Straggler Mitigation in Distributed Learning," *IEEE Information Theory Workshop (ITW)*, 2019.

C2. Rawad Bitar, Yuxuan Xing, Yasaman Keshtkarjahromi, Venkat Dasari, Salim El Rouayheb, and Hulya Seferoglu, "PRAC: Private and Rateless Adaptive Coded Computation at the Edge," *SPIE Defense + Commercial Sensing*, Baltimore, 2019.

C3. Rawad Bitar and Salim El Rouayheb, "Staircase-PIR: Universally Robust Private Information Retrieval," *IEEE Information Theory Workshop (ITW)*, Guangzhou, 2018.

C4. Rawad Bitar, Parimal Parag and Salim El Rouayheb, "Minimizing Latency for Secure Distributed Computing," *IEEE International Symposium on Information Theory (ISIT)*, Aachen, 2017.

C5. Rawad Bitar and Salim El Rouayheb, "Staircase Codes for Secret Sharing with Optimal Communication and Read Overheads," *IEEE International Symposium on Information Theory (ISIT)*, Barcelona, 2016.

P1. Rawad Bitar, Mary Wootters and Salim El Rouayheb, "Stochastic Gradient Coding for Straggler Mitigation in Distributed Learning," submitted to IEEE Journal on Selected Areas in Information Theory, Special Issue on Deep Learning: Mathematical Foundations and Applications to Information Science, 2019.

P2. Rawad Bitar, Yuxuan Xing, Yasaman Keshtkarjahromi, Venkat Dasari, Salim El Rouayheb, and Hulya Seferoglu, "Private and Rateless Adaptive Coded Computation at the Edge," submitted to IEEE Transactions on Information Forensics and Security, 2019.

P3. Rawad Bitar, Parimal Parag and Salim El Rouayheb, "Minimizing Latency for Secure Coded Computing Using Secret Sharing via Staircase Codes," submitted to IEEE Transactions on Communication, 2019.

Chapter 2

Communication Efficient Secret Sharing

We study the communication efficient secret sharing (CE-SS) problem. We use the terminology of secret sharing literature: the workers are called parties, the data of the master is called secret and the master plays the role of a master who shares the secret with the parties and a user who wants to decode the secret. We abstract the concept of stragglers and study the problem of a user contacting a given number of parties to decode the secret.

In the classical setting of threshold secret sharing, a master encodes a secret into n shares and distributes them to n *parties*, such that a *user* contacting any k, $k \leq n$ parties and downloading their shares can decode the secret and any collection of at most z, $z < k$ colluding parties cannot obtain any information about the secret. A CE-SS scheme satisfies the previous properties of threshold secret sharing. Moreover, it allows the user to reconstruct the secret by contacting any set of $d \geq k$, parties, reading and communicating the minimum amount of information. We introduce three explicit constructions of CE-SS codes called *Staircase codes.* The first construction achieves optimal communication and read costs for a given d. The second construction achieves optimal costs universally for all possible values of d between k and n. The

Party 1	Party 2	Party 3	Party 4
$s_1 + s_2 + r_1$	$s_1 + 2s_2 + 4r_1$	$s_1 + 3s_2 + 4r_1$	$s_1 + 4s_2 + r_1$
$r_1 + r_2$	$r_1 + 2r_2$	$r_1 + 3r_2$	$r_1 + 4r_2$

Table 2.1: An example of a CE-SS code based on the Staircase code construction over $\mathbb{F}_5$ for $n = 4$ parties, threshold $k = 2$, $z = 1$ colluding parties and any $d = 3$ parties can efficiently reconstruct the secret. A user contacting any $k = 2$ parties downloads all their shares, i.e., 4 symbols in total, in order to decode the secret. The resulting overheads are $\texttt{CO} = \texttt{RO} = 2$ symbols. However, a user contacting any $d = 3$ parties decodes the secret by downloading the first symbol (in blue) of each share, i.e., 3 symbols in total. Hence, $\texttt{CO} = \texttt{RO} = 1$ symbol. For instance, a user contacting parties 1, 2 and 3 downloads $s_1 + s_2 + r_1$, $s_1 + 2s_2 + 4r_1$, and $s_1 + 3s_2 + 4r_1$ and can decode the secret and r_1. Notice that a user contacting $d = 3$ parties can only decode r_1, whereas a user contacting $k = 2$ parties has to decode r_1 and r_2.

third construction, which is the most general, achieves optimal costs universally for all values of d in any given set $\Delta \subseteq \{k, \dots, n\}$. The introduced Staircase codes allow the master to store a secret of maximal size, i.e., equal to $k - z$ shares, and they are all designed over a small finite field $\mathbb{F}_q$, for any prime power $q > n$. However, Staircase codes may require to divide the secret and the shares into many symbols.

2.1 Introduction

Consider the threshold secret sharing (SS) problem [8, 9] in which a master encodes a secret using random keys into n shares and distributes them to n parties. The threshold SS allows a legitimate user contacting any set of at least k, $k < n$, parties to reconstruct the secret by downloading their shares. In addition, the scheme ensures that any set of at most z, $z < k < n$, colluding parties cannot obtain any information, in an information theoretic sense, about the secret. The following example illustrates the construction of a threshold SS on $n = 4$ shares.

Example 2.1 (Threshold SS). *Let $n = 4, k = 2$ and $z = 1$ and let s be a secret uniformly distributed over $\mathbb{F}_5$. Then, the following 4 shares $(s + r, s + 2r, s + 3r, s + 4r)$ form a threshold SS scheme, with r being a random symbol, called key, chosen uniformly at random from $\mathbb{F}_5$ and independently of s. A user can decode the secret by contacting any $k = 2$ parties, downloading their shares and decoding s and r. Privacy is ensured, because the secret is padded with the key in each share.*

Threshold secret sharing code constructions have been extensively studied in the literature, e.g., [8, 9, 53–57]. The literature on secret sharing predominantly studies non-threshold secret sharing schemes, with so-called general access structures, e.g., [58–60]. We refer the interested reader to the following survey works [61–63] and references within. In this paper, we focus on the problem of communication (and read) efficient secret sharing (CE-SS). A CE-SS scheme satisfies the properties of threshold secret sharing described in the previous paragraph. In addition, it achieves minimum communication and read overheads when the user contacts d parties for any $k \leq d \leq n$. The communication overhead ($\texttt{CO}$) is defined as the extra amount of information (beyond the secret size) downloaded by a user contacting d parties in order to decode the secret. The read overhead $\texttt{RO}$ is defined similarly. Next, we give an example of a CE-SS code that minimizes $\texttt{CO}$ and $\texttt{RO}$. The CE-SS code in this example belongs to the family of Staircase codes which we introduce in Section 2.3.1.

Example 2.2. *Consider again the SS problem of Example 2.1 with $n = 4$, $k = 2$, $z = 1$. We assume now that the secret $\mathbf{s}$ is formed of 2 symbols s_1, s_2 uniformly distributed over $\mathbb{F}_5$ and we use two keys r_1, r_2 drawn independently and uniformly at random from $\mathbb{F}_5$. To construct the Staircase code, the secret symbols and keys are arranged in a matrix M as shown in (2.1). The matrix M is multiplied by a 4×3 Vandermonde matrix V to obtain the matrix $C = VM$. The 4 rows of C form the 4 different shares and give the Staircase*[1] *code shown in Table 2.1.*

$$C = VM = \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 4 \\ 1 & 4 & 1 \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} s_1 & r_1 \\ s_2 & r_2 \\ r_1 & \boxed{0} \end{bmatrix}}_{M}. \tag{2.1}$$

The CE-SS scheme enjoys the following properties. First, a user decodes the secret either by contacting any $k = 2$ parties and downloading all their shares, i.e., 4 symbols, or by contacting any $d = 3$ parties and downloading the first symbol (in blue) of each share, i.e., 3 symbols

[1]The nomenclature of Staircase codes comes from the position of the zero block matrices in the general structure of the matrix M (see the general construction in Table 2.4).

in total. The key idea here is that the user is only interested in decoding the secret and not necessarily the keys. When $d = 3$, the user decodes the secret and only the key r_1, whereas when $d = k = 2$, the user has to decode the secret and both of the keys. This code actually achieves the minimum CO *and* RO *equal to* 1 *symbol for $d = 3$ (and 2 symbols for $d = k = 2$) given later in* (2.4) *and* (2.5)*. Second, privacy is achieved because the secret s_1, s_2 is padded by random keys r_1, r_2 and each $z = 1$ party cannot obtain any information about s_1 and s_2.*

Related work: The CE-SS problem was introduced by Wang and Wong in [22] where they focused on perfect CE-SS, i.e., the case in which $z = k - 1 < n - 1$. The authors showed that there exists a tradeoff between the number of contacted parties d and the amount of information downloaded by a user in order to decode the secret. They derived a lower bound on CO and constructed codes for the special case of $z = k - 1$ using polynomial evaluation over $\mathbb{F}_q$, where $q > n + v$, that achieve minimum CO and RO universally for all $k \leq d \leq k + v - 1$, for some positive integer v. Zhang et al. [64] constructed CE-SS codes for the special case of $z = k - 1$ over $\mathbb{F}_q$, where $q > n$, that achieve minimum CO and RO for any given $k \leq d \leq n$. Recently, Huang et al. [65] studied the CE-SS problem for all values of z, i.e., $z < k < n$, and generalized the lower bound on CO. The authors constructed explicit CE-SS codes for any z achieving the minimum CO and RO for $d = n$ over $\mathbb{F}_q,\ q > n(n - z)$. Moreover, they proved the achievability of the lower bound on CO and RO universally for all possible values of $k \leq d \leq n$ using random linear code constructions[2]. In our setting, we assume that the master has direct access to all the parties. In the case where the master can access the parties through a network, Shah et al. [68] studied the problem of minimizing the communication cost of privately delivering the shares to the parties.

Contributions: We introduce three new classes of explicit constructions of linear CE-SS codes that achieve minimum CO and RO. More specifically, we make the following contributions:

1. We describe the *Staircase code* construction that achieves minimum CO and RO for any

[2]After the appearance of the original version of this work on Arxiv [66], an equivalent CE-SS code construction for all parameters was given concurrently in [67] and [23].

given $k \leq d \leq n$. This construction generalizes the construction in Example 3.1.

2. We describe the *Universal Staircase code* construction that achieves minimum $\texttt{CO}$ and $\texttt{RO}$ simultaneously for all values of $k \leq d \leq n$.

3. We generalize the previous two constructions into the Δ-*Universal Staircase code* construction, which achieves minimum $\texttt{CO}$ and $\texttt{RO}$ simultaneously for all possible values of $d \in \Delta$, for any given set $\Delta \subseteq \{k, \dots, n\}$.

The Staircase codes can store a secret of maximal size, i.e., equal to $k - z$ shares, and require a small finite field $\mathbb{F}_q$ of size $q > n$, which is the same requirement for Reed Solomon based SS codes [53]. However, unlike SS codes, these codes require to divide the secret and the shares into α symbols (α defined later).

Organization: The Chapter is organized as follows. In Section 2.2, we formulate the CE-SS problem, introduce the necessary notations and summarize our results. We describe the Staircase code and Universal Staircase code constructions in Section 2.3. In Section 2.4, we prove that the Staircase codes achieve privacy and minimum $\texttt{CO}$ and $\texttt{RO}$. In Section 2.5, we prove that the Universal Staircase codes achieve privacy and minimum $\texttt{CO}$ and $\texttt{RO}$. We describe the most general Δ-Universal Staircase code construction in Section 2.6. We conclude in Section 4.7.

2.2 Problem Formulation and Main Results

We consider the CE-SS problem. A secret $\mathbf{s}$ of size $u \triangleq k - z$ units is formed of $u\alpha$ symbols (1 unit $= \alpha$ symbols). The secret symbols are drawn independently and uniformly at random from a finite alphabet, typically a finite field. A CE-SS code is a scheme that encodes the secret, using random keys, into n shares $w_1, \dots, w_n$, of unit size each, and distributes them to n distinct parties. Let $\texttt{W}_i$ denote the random variable representing the share of party i, let $\texttt{S}$ denote the random variable representing the secret $\mathbf{s}$, let $[n] = \{1, \dots, n\}$, and for any subset $\mathcal{B} \subseteq [n]$ denote by $\texttt{W}_{\mathcal{B}}$ the set of random variables representing the shares indexed by $\mathcal{B}$, i.e., $\texttt{W}_{\mathcal{B}} = \{\texttt{W}_i; i \in \mathcal{B}\}$. Then, a CE-SS code must satisfy the following properties:

1. *Perfect privacy:* Any subset of z or less parties should not be able to get any information about the secret. The perfect privacy condition can be expressed as

$$H(\mathtt{S} \mid \mathtt{W}_{\mathcal{Z}}) = H(\mathtt{S}), \forall \mathcal{Z} \subset [n] \text{ s.t. } |\mathcal{Z}| = z. \tag{2.2}$$

2. *MDS:* A user downloading k shares is able to recover the secret, i.e.,

$$H(\mathtt{S} \mid \mathtt{W}_{\mathcal{A}}) = 0, \forall \mathcal{A} \subseteq [n] \text{ s.t. } |\mathcal{A}| = k, \tag{2.3}$$

   and the secret is of maximal size $u = k - z$ units as implied by (2.2) and (2.3) (see [65, Proposition 1]).

3. *Minimum $\mathtt{CO}$ and $\mathtt{RO}$:* a user contacting d parties, $k \leq d \leq n$, is able to decode the secret by reading and downloading exactly $u + \mathtt{CO}(d)$ units of information in total from all the contacted shares, where

$$\mathtt{CO}(d) = \frac{uz}{d-z}. \tag{2.4}$$

Equation (2.4) represents the achievable information theoretic lower bound [22, Theorem 3.1], [65, Theorem 1] on the communication overhead, $\mathtt{CO}(d)$, needed to satisfy the constraints in (2.2) and (2.3), when the user contacts d parties[3]. Since the amount of information read cannot be less than the downloaded amount, the following lower bound on $\mathtt{RO}$ holds,

$$\mathtt{RO}(d) \geq \mathtt{CO}(d). \tag{2.5}$$

We will refer to a CE-SS code described above as an (n, k, z, d) CE-SS code. For instance, the code in Example 3.1 is an $(4, 2, 1, 3)$ CE-SS code. We define a universal (n, k, z) CE-SS code that achieves minimum $\mathtt{CO}(d)$ and $\mathtt{RO}(d)$ simultaneously for all possible values of d. Note that

[3]Note that a user contacting d parties and achieving (2.4) for a threshold secret sharing with threshold k downloads the same amount of information as a user contacting d parties in a threshold secret sharing with threshold d.

the MDS constraint can be omitted since it is subsumed by the minimum $\texttt{CO}$ and $\texttt{RO}$ constraint since it corresponds to the case of $d = k$ and $\texttt{CO}(k) = z$. However, we will make this distinction for clarity of exposition. In the sequel whenever referring to k, $k < n$ should be understood and whenever referring to z, $z < k < n$ should be understood. Also, whenever referring to d, $k \leq d \leq n$ should be understood unless otherwise stated.

Given the model described above, we are ready to state our main results. We introduce the first class of Staircase codes that satisfy minimum $\texttt{CO}$ and $\texttt{RO}$ for any given d.

Theorem 2.1. *The (n, k, z, d) Staircase CE-SS code defined in Section 2.3.1 over $\mathbb{F}_q$, $q > n$, satisfies the required MDS and perfect privacy constraints given in (2.2) and (2.3) for any given $z < k < n$, and achieves optimal communication and read overheads $\texttt{CO}(d)$ and $\texttt{RO}(d)$ given in (2.4) and (2.5) for any given $k \leq d \leq n$.*

Our next result introduces the Universal Staircase codes which achieve optimal overheads simultaneously for all possible values of d.

Theorem 2.2. *The (n, k, z) Universal Staircase CE-SS code defined in Section 2.3.2 over $\mathbb{F}_q$, $q > n$, satisfies the required MDS and perfect privacy constraints given in (2.2) and (2.3) for any given $z < k < n$, and achieves optimal communication and read overheads $\texttt{CO}(d)$ and $\texttt{RO}(d)$ given in (2.4) and (2.5) simultaneously for all $k \leq d \leq n$.*

Theorem 2.3 generalizes the first two constructions.

Theorem 2.3. *Let $\Delta \subseteq \{k, \ldots, n\}$. The (n, k, z, Δ) Δ-universal Staircase codes defined in Section 2.6 over $\mathbb{F}_q$, $q > n$, satisfies the required MDS and perfect privacy constraints given in (2.2) and (2.3) for any given $z < k < n$, and achieves optimal communication overhead $\texttt{CO}(d)$ and read overhead $\texttt{RO}(d)$ given in (2.4) and (2.5) simultaneously for all $d \in \Delta$.*

All Staircase codes constructions require dividing the shares into α symbols (therefore dividing the secret into $u\alpha$ symbols) as defined in Sections 2.3 and 2.6. No lower bounds on α are currently known for a given field size q.

Symbol	Meaning	Symbol	Meaning
n	number of parties	$\texttt{RO}$	read overhead
k	threshold on contacted parties	$\texttt{CO}$	communication overhead
z	number of colluding parties	V	Vandermonde matrix
d	number of contacted parties	S	matrix containing the secret
$\mathbf{s}$	secret of size $k-z$ units	R_i	matrix containing random symbols
α	number of symbols in one unit	D_j	matrix with entries from S and R_i, $i \leq j$
r	random symbol	M	structured matrix containing S, R_i's and D_j's
w	share of unit size given to a party	$[n]$	set of natural numbers from 1 to n
h	defined as $h \triangleq n-k+1$	$\mathcal{Z}$	$\mathcal{Z} \subset [n]$, $\lvert\mathcal{Z}\rvert = z$, indices of colluding parties
d_i	defined as $d_i \triangleq n-i+1$	$\mathcal{I}$	$\mathcal{I} \subset [n]$, $\lvert\mathcal{I}\rvert = d$, indices of contacted parties
α_i	defined as $\alpha \triangleq d_i - z$	$V_{\mathcal{Z}}$	matrix formed of rows of V indexed by $\mathcal{Z}$
α	defined as $\alpha \triangleq \text{LCM}(\alpha_1, \ldots, \alpha_{h-1})$	$\texttt{S}$	random variable representing the secret $\mathbf{s}$
u	secret size $u \triangleq k-z$ units	$\texttt{W}$	random variable representing the share w
$\mathbf{0}$	matrix with all 0 entries	$\texttt{R}$	random variable representing all the keys

Table 2.2: Summary of notations for this Chapter.

We note that codes achieving minimum $\texttt{CO}$ and $\texttt{RO}$ for all possible values of d were presented in [67] and [23] concurrently with the original version of this work [66]. As explained in [23], the Staircase code constructions and the construction in [23] are equivalent. The equivalence between the two constructions is similar to the equivalence between the construction of threshold SS codes based on linear block codes and the one based on polynomial evaluations[4]. Staircase codes presented here is a linear block code construction based on Vandermonde matrices, whereas the construction presented in [23] is based on polynomial evaluations.

2.3 Staircase Code Constructions

2.3.1 Staircase code construction for given d

We describe the (n, k, z, d) Staircase code construction that achieves optimal communication and read overheads $\texttt{CO}(d)$ and $\texttt{RO}(d)$ for any given $k \leq d \leq n$. In this construction, we take $\alpha = d - z$. Hence, the secret $\mathbf{s}$ of size u units is formed of $u(d-z)$ symbols $s_1, \ldots, s_{u\alpha}$, where $s_i \in \mathbb{F}_q$ and $q > n$. The symbols s_i are arranged in an $\alpha \times u$ matrix S. The construction uses $z\alpha$ iid random keys drawn uniformly at random from $\mathbb{F}_q$ and independently of the secret. The keys

[4]This is also similar to constructing Reed Solomon codes either based on polynomial evaluations, or based on linear block codes with a Vandermonde generator matrix.

are partitioned into two matrices R_1 and R_2 of dimensions $z \times u$ and $z \times (\alpha - u)$ respectively. Let D be the transpose of the last $(\alpha - u)$ rows of the matrix[5] $\begin{bmatrix} S \\ R_1 \end{bmatrix}$ and let $\mathbf{0}$ be the all zero square matrix of dimensions $(\alpha - u) \times (\alpha - u)$, note that $\alpha - u \geq 0$ since $d \geq z + u$. The key ingredient of the construction is to arrange the secret and the keys in a $d \times \alpha$ matrix M defined in Table 2.3. The inspiration behind this construction is the class of Product Matrix codes that minimizes the repair bandwidth in distributed storage systems[6] [70].

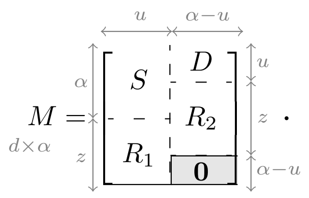


Table 2.3: The structure of the matrix M that contains the secret and keys in the Staircase code construction for given d.

Encoding: Let V be an $n \times d$ Vandermonde[7] matrix defined over $\mathbb{F}_q$. The matrix M, defined in Table 2.3, is multiplied by V to obtain the matrix $C = VM$. The n rows of C form the n different shares.

Decoding: A user contacting k parties downloads all the shares of the contacted parties. A user contacting d parties, indexed by $\mathcal{I} \subseteq [n]$, downloads the first k symbols from each contacted party corresponding to $v_i \begin{bmatrix} S & R_1 \end{bmatrix}^T$, $i \in \mathcal{I}$, where v_i denotes the i^{th} row of the Vandermonde matrix V (the superscript T denotes the transpose of a matrix). The decoding procedure given in the proof of Theorem 2.1 guarantees that the user will be able to decode the secret in both cases.

Example 2.2 (Continued)**.** *We give the details of the construction of the* $(n, k, z, d) = (4, 2, 1, 3)$ *CE-SS code of Example 3.1. Recall that* $u = k - z = 1$. *We take* $\alpha = d - z = 2$, *thus the secret* $\mathbf{s}$ *is formed of* $u\alpha = 2$ *symbols* $s_1,\ s_2$ *uniformly distributed over* $\mathbb{F}_q$, $q = 5 > n = 4$. *The construction*

[5]If $\alpha - u \leq z$, i.e., $d \leq 2z + u$, then D consists of the transpose of the last $\alpha - u$ rows of R_1.

[6]After the appearance of the original version of this work on Arxiv [66], a connection between the family of regenerating codes and CE-SS codes was explored in more details in [69].

[7]We require all square sub-matrices formed by consecutive columns of V to be invertible. Vandermonde and Cauchy matrices satisfy this property.

uses $z\alpha = 2$ iid random keys $r_1,\ r_2$ drawn uniformly at random over $\mathbb{F}_5$ and independently of the secret. The keys are partitioned into two matrices R_1 and R_2 of dimensions $z \times u = 1 \times 1$ and $z \times (\alpha - u) = 1 \times 1$, respectively. The matrix D is the transpose of the last $\alpha - u = 1$ row of R_1. Hence, we have, $R_1 = D = r_1,\ R_2 = r_2$, and $S = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$. The secret and the keys are arranged in a $d \times \alpha = 3 \times 2$ matrix M. Let V be an $n \times d = 4 \times 3$ Vandermonde matrix. M and V are given again in (2.6).

$$M = \begin{bmatrix} s_1 & r_1 \\ s_2 & r_2 \\ r_1 & 0 \end{bmatrix} \text{ and } V = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 4 \\ 1 & 4 & 1 \end{bmatrix}. \tag{2.6}$$

The shares are the rows of the matrix $C = VM$ as given in Table 2.1. We want to check that this code satisfies the following properties:

1) Minimum CO and RO for $d = 3$: *We check that a user contacting $d = 3$ parties can reconstruct the secret with minimum CO and RO. For instance, if a user contacts the first 3 parties and downloads the first symbol of each contacted share, then the downloaded data is given by,*

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 4 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ r_1 \end{bmatrix}. \tag{2.7}$$

The matrix on the left is a 3×3 square Vandermonde matrix, hence invertible. Therefore, the user can decode the secret (and r_1). This remains true irrespective of which 3 parties are contacted. The user reads and downloads 3 symbols of size $3/\alpha = 3/2$ units resulting in minimum overheads, $\text{CO}(3) = \text{RO}(3) = 3/2 - u = 1/2$, as given in (2.4) and (2.5).

2) MDS: *We check that a user contacting $k = 2$ parties can reconstruct the secret. Suppose the user contacts parties 1 and 2 and downloads all their shares given by*

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} s_1 & r_1 \\ s_2 & r_2 \\ r_1 & 0 \end{bmatrix}. \tag{2.8}$$

The system in (2.8) is equivalent to the two following systems $\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 4\end{bmatrix}\begin{bmatrix}s_1\\ s_2\\ r_1\end{bmatrix}$ *and* $\begin{bmatrix}1 & 1\\ 1 & 2\end{bmatrix}\begin{bmatrix}r_1\\ r_2\end{bmatrix}$. *The decoder uses the latter system to decode r_1 and r_2. This is possible because the matrix on the left is a square Vandermonde matrix, hence invertible. Then, the decoder subtracts the obtained value of r_1 from the former system to obtain again the following invertible system* $\begin{bmatrix}1 & 1\\ 1 & 2\end{bmatrix}\begin{bmatrix}s_1\\ s_2\end{bmatrix}$. *The decoder then decodes s_1 and s_2. Again, this procedure is possible for any 2 contacted parties.*

3) Perfect privacy: *At a high level, perfect privacy is achieved here because each symbol in a share is "padded" with at least one distinct key statistically independent of the secret, making the shares of any party independent of the secret.*

2.3.2 Universal Staircase code construction

We describe the (n, k, z) Universal Staircase code construction that achieves optimal communication and read overheads $\texttt{CO}(d)$ and $\texttt{RO}(d)$ simultaneously for all possible values of $k \le d \le n$. Let $d_1 = n, d_2 = n-1, \dots, d_h = k$, with $h = n-k+1$, and $\alpha_i = d_i - z,\ i = 1, \dots, h$. Choose $\alpha = \text{LCM}(\alpha_1, \alpha_2, \dots, \alpha_{h-1})$, that is the least common multiple of all the α_i's except for the last $\alpha_h = k - z = u$. The secret $\mathbf{s}$ consists of $u\alpha$ symbols $s_1, \dots, s_{u\alpha}$, uniformly distributed over $\mathbb{F}_q,\ q > n$, arranged in an $\alpha_1 \times u\alpha/\alpha_1$ matrix S.

The construction uses $z\alpha$ iid random keys, drawn uniformly at random from $\mathbb{F}_q$ and independently of the secret. The keys are partitioned into h matrices $R_i, i = 1, \dots, h$, of respective dimensions $z \times u\alpha/\alpha_i\alpha_{i-1}$ (take $\alpha_0 = 1$). The matrices $R_1, \dots, R_i$ consist of the overhead of keys decoded by a user contacting d_i parties. We form h matrices M_i, $i = 1, \dots, h$, as follows,

$$M_1 = n\underbrace{\begin{bmatrix} S \\ R_1 \end{bmatrix}}_{u\alpha/\alpha_1}\begin{matrix}\alpha_1\\ z\end{matrix}, \quad M_2 = n\underbrace{\begin{bmatrix} D_1 \\ R_2 \\ \mathbf{0} \end{bmatrix}}_{u\alpha/\alpha_1\alpha_2}\begin{matrix}\alpha_2\\ z\\ 1\end{matrix}, \ \dots, \quad M_j = n\underbrace{\begin{bmatrix} D_{j-1} \\ R_j \\ \mathbf{0} \end{bmatrix}}_{u\alpha/\alpha_{j-1}\alpha_j}\begin{matrix}\alpha_j\\ z\\ n-d_j\end{matrix}, \ \dots, \quad M_h = n\underbrace{\begin{bmatrix} D_{h-1} \\ R_h \\ \mathbf{0} \end{bmatrix}}_{\alpha/\alpha_{h-1}}\begin{matrix}u\\ z\\ h-1\end{matrix}. \tag{2.9}$$

Each matrix D_j is formed of the $(n-j+1)^{th}$ row of $\begin{bmatrix}M_1 & M_2 \dots M_j\end{bmatrix}$ wrapped around to make

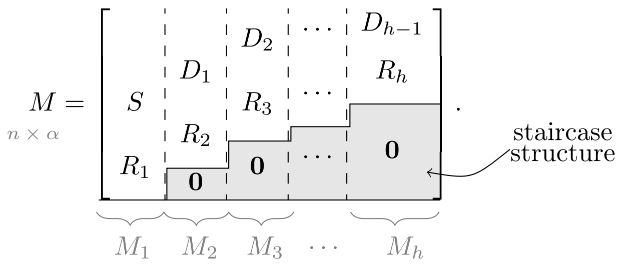


Table 2.4: Structure of the matrix M that contains the secret and keys in the universal Staircase code construction.

a matrix of dimensions $\alpha_{j+1} \times u\alpha/\alpha_j\alpha_{j+1}$ for $j = 1, \dots, h-1$. The $\mathbf{0}$'s are the all zero matrices used to complete the M_i's to n rows. The secret and the keys are arranged in the matrix $M = \begin{bmatrix} M_1 \dots M_h \end{bmatrix}$ defined in Table 2.4.

The matrix M is characterized by a special structure resulting from carefully choosing the entries of the D_j's and placing the all zero sub-blocks in a staircase shape, giving these codes their name. This staircase shape allows to achieve optimal communication and read overheads $\texttt{CO}$ and $\texttt{RO}$ for all d.

Encoding: The encoding is similar to the Staircase code construction. Let V be an $n \times n$ Vandermonde matrix defined over $\mathbb{F}_q$. The matrix M, defined in Table 2.4, is multiplied by V to obtain the matrix $C = VM$. The n rows of C form the n different shares.

Decoding: To reconstruct the secret, a user contacting d_j parties indexed by $\mathcal{I} \subseteq [n]$ downloads the first $u\alpha/\alpha_j$ symbols from each contacted party corresponding to $v_i \begin{bmatrix} M_1 \dots M_j \end{bmatrix}$, for all $i \in \mathcal{I}$. The decoding procedure given in the proof of Theorem 2.2 guarantees that the user will be able to decode the secret.

We postpone the example of a Universal Staircase code to Section 2.5.1 to have it next to the proof of Theorem 2.2.

2.4 Staircase Code for Given d

Proof of Theorem 2.1. Consider the (n, k, z, d) Staircase code defined in Section 2.3.1 with $u = k - z$. We prove Theorem 2.1 by establishing the following properties of the code:

1) Minimum $\texttt{CO}(d)$ *and* $\texttt{RO}(d)$*:* We prove that a user contacting d parties can reconstruct the secret while incurring minimum $\texttt{CO}$ and $\texttt{RO}$. A user contacting d parties downloads the first u symbols of each party. Let $\mathcal{I} \subset [n]$, $|\mathcal{I}| = d$, be the set of indices of the contacted parties, then the downloaded data is given by $V_{\mathcal{I}} \begin{bmatrix} S & R_1 \end{bmatrix}^T$, where $V_{\mathcal{I}}$ is a $d \times d$ square Vandermonde matrix formed of the rows of V indexed by $\mathcal{I}$, hence invertible. The user can always decode the secret (and the keys in R_1) by inverting $V_{\mathcal{I}}$. The code is optimal on communication and read overheads $\texttt{CO}(d)$ and $\texttt{RO}(d)$, because the user only reads and downloads ud symbols of size $ud/\alpha = ud/(d-z)$ units resulting in an overhead of $ud/\alpha - u = uz/\alpha = uz/(d-z)$ achieving the optimal $\texttt{CO}(d)$ and $\texttt{RO}(d)$ given in (2.4) and (2.5).

2) MDS: We prove that a user contacting k parties and downloading all their shares can reconstruct the secret. Let $\mathcal{I} \subset [n], |\mathcal{I}| = k$, be the set of indices of the contacted parties. The information downloaded by the user is $V_{\mathcal{I}}M$ and is given by,

$$V_{\mathcal{I}} \begin{bmatrix} S & D \\ & R_2 \\ R_1 & \\ & \mathbf{0} \end{bmatrix}.$$

First, we show that the user can decode the entries of D and R_2. The decoder considers the system,

$$V_{\mathcal{I}} \begin{bmatrix} D & R_2 & \mathbf{0} \end{bmatrix}^T = V'_{\mathcal{I}} \begin{bmatrix} D & R_2 \end{bmatrix}^T. \tag{2.10}$$

Recall that the dimensions of the all zero matrix in (2.10) are $(\alpha - u) \times (\alpha - u)$, then $V'_{\mathcal{I}}$ is a $k \times k$ square Vandermonde matrix formed by the first k columns of $V_{\mathcal{I}}$. Therefore, the user can always decode the entries of D and R_2 because $V'_{\mathcal{I}}$ is invertible. Second, we prove that the user can always decode the entries of S and R_1 and hence reconstruct the secret. Recall that D is the transpose of the last $\alpha - u$ rows of $M_1 \triangleq \begin{bmatrix} S & R_1 \end{bmatrix}^T$. By subtracting the previously decoded entries of D from $V_{\mathcal{I}} \begin{bmatrix} S & R_1 \end{bmatrix}^T$, the user obtains $V'_{\mathcal{I}} M'_1$, where $V'_{\mathcal{I}}$ is defined above and M'_1 is a $k \times u$ matrix formed by the first k rows of M_1. Therefore, the user can always decode the entries of M'_1 because $V'_{\mathcal{I}}$ is invertible. If $k \geq \alpha$, then S is directly obtained since it is contained in M_1'. Otherwise, M_1' consists of the first k rows of S. The remaining rows of S are contained in D and were previously decoded. In both cases, the user can decode all the secret symbols $s_1, \ldots, s_{u\alpha}$.

3) Perfect privacy: We prove that for any subset $\mathcal{Z} \subset [n]$, $|\mathcal{Z}| = z$, the collection of shares indexed by z, denoted by $W_{\mathcal{Z}} = \{w_i, i \in \mathcal{Z}\}$, does not reveal any information about the secret as given in equation (2.2), i.e., $H(\mathtt{S} \mid \mathtt{W}_{\mathcal{Z}}) = H(\mathtt{S})$ where $\mathtt{W}_{\mathcal{Z}}$ denotes the set of random variable representing the collection of shares $W_{\mathcal{Z}}$. Let $\mathtt{R}$ denote the random variable representing all the random keys, then it suffices to prove that $H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}) = 0$ as detailed in Lemma 2.4.

Lemma 2.4. *Let $\mathtt{W}_i$ denote the random variable representing share w_i, and for any subset $\mathcal{B} \subseteq \{1, \ldots, n\}$ denote by $\mathtt{W}_{\mathcal{B}}$ the set random variables representing the shares indexed by $\mathcal{B}$, i.e., $\mathtt{W}_{\mathcal{B}} = \{\mathtt{W}_i; i \in \mathcal{B}\}$. For all $\mathcal{Z} \subset \{1, \ldots, n\}$, $|\mathcal{Z}| = z$, the perfect privacy constraint $H(\mathtt{S} \mid \mathtt{W}_{\mathcal{Z}}) = H(\mathtt{S})$, given in (2.2), is equivalent to $H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}) = 0$.*

Proof. The proof is standard [71,72] but we reproduce it here for completeness. In what follows, the logarithms in the entropy function are taken base q. We can write,

$$\begin{aligned}
H(\mathtt{S} \mid \mathtt{W}_{\mathcal{Z}}) &= H(\mathtt{S}) - H(\mathtt{W}_{\mathcal{Z}}) + H(\mathtt{W}_{\mathcal{Z}} \mid \mathtt{S}) \\
&= H(\mathtt{S}) - H(\mathtt{W}_{\mathcal{Z}}) + H(\mathtt{W}_{\mathcal{Z}} \mid \mathtt{S}) - H(\mathtt{W}_{\mathcal{Z}} \mid \mathtt{S}, \mathtt{R}) && (2.11) \\
&= H(\mathtt{S}) - H(\mathtt{W}_{\mathcal{Z}}) + I(\mathtt{W}_{\mathcal{Z}}; \mathtt{R} \mid \mathtt{S}) \\
&= H(\mathtt{S}) - H(\mathtt{W}_{\mathcal{Z}}) + H(\mathtt{R} \mid \mathtt{S}) - H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}) \\
&= H(\mathtt{S}) - H(\mathtt{W}_{\mathcal{Z}}) + H(\mathtt{R}) - H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}) && (2.12) \\
&= H(\mathtt{S}) - z\alpha + z\alpha - H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}) && (2.13) \\
&= H(\mathtt{S}) - H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}). && (2.14)
\end{aligned}$$

Equation (C.2) follows from the fact that given the secret $\mathbf{s}$ and the keys R any share can be decoded, equation (2.12) follows because the random keys are chosen independently from the secret and equation (C.6) follows because the Staircase code constructions use $z\alpha$ independent random keys.

Therefore, to prove that the privacy condition $H(\mathtt{S} \mid \mathtt{W}_{\mathcal{Z}}) = H(\mathtt{S})$ is satisfied is equivalent to prove that $H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}) = 0$. $\square$

Therefore, we need to show that given the secret $\mathbf{s}$ as side information, any collection of z shares can decode all the random keys. A collection of $W_{\mathcal{Z}}$ shares can be written as

$$V_{\mathcal{Z}} \begin{bmatrix} S & D \\ & R_2 \\ R_1 & \\ & \mathbf{0} \end{bmatrix}, \tag{2.15}$$

where V_Z is a $z \times d$ matrix corresponding to the rows of V_Z indexed by Z. The linear system in (2.15) can be divided into two systems as follows,

$$V_{\mathcal{Z}} \begin{bmatrix} S & R_1 \end{bmatrix}^T, \tag{2.16}$$

$$V_{\mathcal{Z}} \begin{bmatrix} D & R_2 & \mathbf{0} \end{bmatrix}^T. \tag{2.17}$$

Given the secret as side information, it can be subtracted from (2.16), which becomes

$$V_{\mathcal{Z}} \begin{bmatrix} \mathbf{0} & R_1 \end{bmatrix}^T = V''_{\mathcal{Z}} R_1,$$

where, $V''_{\mathcal{Z}}$ is a $z \times z$ square Vandermonde matrix consisting of the last z columns of $V_{\mathcal{Z}}$. The entries of R_1 can always be decoded because $V''_{\mathcal{Z}}$ is invertible. Now that R_1 is decoded and we have S as side information, we can obtain D as the last $\alpha - u$ rows of $\begin{bmatrix} S & R_1 \end{bmatrix}^T$. Then, the entries of D are subtracted from the second system to obtain $V^*_{\mathcal{Z}} R_2$, where $V^*_{\mathcal{Z}}$ is a $z \times z$ square Vandermonde matrix consisting of the $(u+1)^{\text{st}}$ to the k^{th} columns of $V_{\mathcal{Z}}$. Hence, the entries of R_2 can always be decoded because $V^*_{\mathcal{Z}}$ is invertible. Therefore, $H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}) = 0$, $\forall\ \mathcal{Z},\ \mathcal{Z} \subset [n],\ |\mathcal{Z}| = z$ and perfect privacy is achieved.

$\square$

Party 1	Party 2	Party 3	Party 4
$s_1 + s_2 + s_3 + r_1$	$s_1 + 2s_2 + 4s_3 + 3r_1$	$s_1 + 3s_2 + 4s_3 + 2r_1$	$s_1 + 4s_2 + s_3 + 4r_1$
$s_4 + s_5 + s_6 + r_2$	$s_4 + 2s_5 + 4s_6 + 3r_2$	$s_4 + 3s_5 + 4s_6 + 2r_2$	$s_4 + 4s_5 + s_6 + 4r_2$
$r_1 + r_2 + r_3$	$r_1 + 2r_2 + 4r_3$	$r_1 + 3r_2 + 4r_3$	$r_1 + 4r_2 + r_3$
$s_3 + r_4$	$s_3 + 2r_4$	$s_3 + 3r_4$	$s_3 + 4r_4$
$s_6 + r_5$	$s_6 + 2r_5$	$s_6 + 3r_5$	$s_6 + 4r_5$
$r_3 + r_6$	$r_3 + 2r_6$	$r_3 + 3r_6$	$r_3 + 4r_6$

Table 2.5: An example of a universal Staircase code for $(n, k, z) = (4, 2, 1)$ over $\mathbb{F}_5$.

2.5 Universal Staircase Codes

2.5.1 Example

We describe here the construction of an $(n, k, z) = (4, 2, 1)$ Universal Staircase code over $\mathbb{F}_q$, $q = 5 > n = 4$, by following the construction in Section 2.3.2. We have $d_1 = 4$, $d_2 = 3$, $d_3 = 2$ and $\alpha_1 = 3$, $\alpha_2 = 2$, $\alpha_3 = 1$ and $\alpha = \text{LCM}(\alpha_1, \alpha_2) = \text{LCM}(3, 2) = 6$. The secret $\mathbf{s}$ is formed of $u\alpha = (k - z)\alpha = 6$ symbols uniformly distributed over $\mathbb{F}_5$. The construction uses $z\alpha = 6$ iid random keys drawn uniformly at random from $\mathbb{F}_5$ and independently of the secret. The secret symbols and the random keys are arranged in the following matrices,

$$S = \begin{bmatrix} s_1 & s_4 \\ s_2 & s_5 \\ s_3 & s_6 \end{bmatrix}, \quad R_1 = \begin{bmatrix} r_1 & r_2 \end{bmatrix},$$

$$R_2 = \begin{bmatrix} r_3 \end{bmatrix} \quad \text{and} \quad R_3 = \begin{bmatrix} r_4 & r_5 & r_6 \end{bmatrix}.$$

To build the matrix M which will be used for encoding the secret, we start with

$$M_1 = \begin{bmatrix} \\ S \\ \\ R_1 \end{bmatrix} = \begin{bmatrix} s_1 & s_4 \\ s_2 & s_5 \\ s_3 & s_6 \\ r_1 & r_2 \end{bmatrix}.$$

Then, D_1 is the $\alpha_2 \times u\alpha/\alpha_1\alpha_2 = 2 \times 1$ matrix that contains the symbols of the n^{th} row of M_1, i.e., $D_1 = \begin{bmatrix} r_1 & r_2 \end{bmatrix}^T$. Therefore, $M_2 = \begin{bmatrix} D_1 & R_2 & \mathbf{0} \end{bmatrix}^T = \begin{bmatrix} r_1 & r_2 & r_3 & 0 \end{bmatrix}^T$. Similarly, we have

$D_2 = \begin{bmatrix} s_3 & s_6 & r_3 \end{bmatrix}$ and

$$M_3 = \begin{bmatrix} s_3 & s_6 & r_3 \\ r_4 & r_5 & r_6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

We obtain M by concatenating $M_1,\ M_2$ and M_3,

$$M = \begin{bmatrix} \underbrace{\begin{matrix} s_1 & s_4 \\ s_2 & s_5 \\ s_3 & s_6 \\ r_1 & r_2 \end{matrix}}_{M_1} & \underbrace{\begin{matrix} r_1 \\ r_2 \\ r_3 \\ 0 \end{matrix}}_{M_2} & \underbrace{\begin{matrix} s_3 & s_6 & r_3 \\ r_4 & r_5 & r_6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}}_{M_3} \end{bmatrix}. \tag{2.18}$$

Here, V is the $n \times n = 4 \times 4$ Vandermonde matrix over $\mathbb{F}_5$ given in (2.19). The shares are given by the rows of the matrix $C = VM$ and shown in Table 2.5.

$$V = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \end{bmatrix}. \tag{2.19}$$

The constructed Universal Staircase code satisfies the following properties:

1) MDS: We check that a user contacting $d_3 = k = 2$ parties can decode the secret. Suppose that the user contacts parties 1 and 2. The data downloaded by the user is $V_{\{1,2\}}M$ and is given by,

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \end{bmatrix}}_{V_{\{1,2\}}} \begin{bmatrix} \underbrace{\begin{matrix} s_1 & s_4 \\ s_2 & s_5 \\ s_3 & s_6 \\ r_1 & r_2 \end{matrix}}_{M_1} & \underbrace{\begin{matrix} r_1 \\ r_2 \\ r_3 \\ 0 \end{matrix}}_{M_2} & \underbrace{\begin{matrix} s_3 & s_6 & r_3 \\ r_4 & r_5 & r_6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}}_{M_3} \end{bmatrix}. \tag{2.20}$$

We will show that the user can decode the secret by successively solving the linear systems $V_{\{1,2\}}M_3$, $V_{\{1,2\}}M_2$ and $V_{\{1,2\}}M_1$. The decoder starts by considering $V_{\{1,2\}}M_3$ which gives,

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} s_3 & s_6 & r_3 \\ r_4 & r_5 & r_6 \end{bmatrix}. \tag{2.21}$$

The matrix on the left is invertible, and the user can decode the secret symbols and keys in (2.21). Then, the decoder considers the system $V_{\{1,2\}}M_2$ after subtracting from it the value of r_3 decoded in the previous step. The obtained system is again invertible and the decoder can decode r_1 and r_2. The decoder then considers $V_{\{1,2\}}M_1$, after canceling out $r_1,\ r_2,\ s_3,\ s_6$ decoded so far, to obtain the following system,

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} s_1 & s_4 \\ s_2 & s_5 \end{bmatrix}.$$

The matrix on the left is again invertible and the decoder can reconstruct the secret. This remains true irrespective of which 2 parties are contacted.

2) Minimum $\texttt{CO}$ *and* $\texttt{RO}$ *for* $d_2 = 3$ *and* $d_1 = 4$*:* We check that a user contacting d parties, $d = 3, 4$, can decode the secret while achieving the minimum communication and read overheads given in (2.4) and (2.5). Suppose a user contacts $d_2 = 3$ parties indexed by $\mathcal{I} \subset [n]$. The user reads and downloads the first $u\alpha/\alpha_2 = 3$ symbols of each contacted share corresponding to $V_{\mathcal{I}} \begin{bmatrix} M_1 & M_2 \end{bmatrix}$ (in black and red), where $V_{\mathcal{I}}$ is the matrix formed by the rows of V indexed by $\mathcal{I}$. The user will be able to reconstruct the secret by implementing a decoding procedure similar to the one above. The resulting $\texttt{CO}$ and $\texttt{RO}$ are equal to $3/2 - u = 1/2$ units achieving the optimal $\texttt{CO}(d_2)$ and $\texttt{RO}(d_2)$ given in (2.4) and (2.5). In the case when a user contacts $d_1 = 4$ parties, the user reads and downloads the first $u\alpha/\alpha_1 = 2$ symbols of each contacted share corresponding to $V_{\mathcal{I}}M_1$ (in black). The user can always decode the secret because $V_{\mathcal{I}}$ here is a 4×4 square Vandermonde matrix, hence invertible. The resulting $\texttt{CO}$ and $\texttt{RO}$ are equal to $1/3$ achieving the optimal $\texttt{CO}(d_1)$ and $\texttt{RO}(d_1)$ given in (2.4) and (2.5).

3) Perfect privacy: At a high level, perfect privacy is achieved here because each symbol in a share is "padded" with at least one distinct key statistically independent of the secret, making the shares of any party independent of the secret.

2.5.2 Proof of Theorem 2.2

Consider the (n, k, z) Universal Staircase code construction defined in Section 2.3.2. We prove Theorem 2.2 by establishing the following properties.

1) Encoding is well defined: We prove that the $(n-j+1)^{\text{st}}$ row of $\begin{bmatrix} M_1 \dots M_j \end{bmatrix}$ has the same number of entries as $D_j, j = 1, \dots, h-1$. Therefore, we can always construct the matrix D_j. In fact, the number of entries of one row of $\begin{bmatrix} M_1 \dots M_j \end{bmatrix}$ is equal to the sum of the number of columns of the M_i's, $i = 1, \dots, j$. Notice that $\alpha_{i-1} = \alpha_i + 1$, then we can write,

$$\frac{u\alpha}{\alpha_i \alpha_{i-1}} = u\alpha \left(\frac{1}{\alpha_i} - \frac{1}{\alpha_{i-1}} \right).$$

Hence, the number of columns of $\begin{bmatrix} M_1 \dots M_j \end{bmatrix}$ is given by,

$$\frac{u\alpha}{\alpha_1} + u\alpha \left(\frac{1}{\alpha_2} - \frac{1}{\alpha_1} \right) + \dots + u\alpha \left(\frac{1}{\alpha_j} - \frac{1}{\alpha_{j-1}} \right) = \frac{u\alpha}{\alpha_j}, \tag{2.22}$$

which is equal to the number of entries of D_j.

2) MDS and minimum CO(d) *and* RO(d) *for all* $k \leq d \leq n$*:* We prove that for all $k \leq d \leq n$, a user contacting d parties can decode the secret while achieving the minimum communication and read overheads given in (2.4) and (2.5). Notice that the MDS property follows directly from the fact that a user contacting $d_h = k$ parties can reconstruct the secret by reading and downloading all the contacted shares.

A user contacting d_j, $j = 1, \dots, h$, parties downloads the first $u\alpha/\alpha_j$ symbols of each party. Let $\mathcal{I} \subseteq [n]$, $|\mathcal{I}| = d_j$, be the set of indices of the contacted parties and let $V_{\mathcal{I}}$ be the matrix formed of the rows of V indexed by $\mathcal{I}$. The total downloaded data is given by $V_{\mathcal{I}} \begin{bmatrix} M_1 \dots M_j \end{bmatrix}$

and can be divided into j linear systems given as follows,

$$V_{\mathcal{I}} M_1 = V_{\mathcal{I}} \begin{bmatrix} S & R_1 \end{bmatrix}^T \tag{2.23}$$

$$V_{\mathcal{I}} M_2 = V_{\mathcal{I}} \begin{bmatrix} D_1 & R_2 & \mathbf{0} \end{bmatrix}^T \tag{2.24}$$

$$\vdots$$

$$V_{\mathcal{I}} M_{j-1} = V_{\mathcal{I}} \begin{bmatrix} D_{j-2} & R_{j-1} & \mathbf{0} \end{bmatrix}^T \tag{2.25}$$

$$V_{\mathcal{I}} M_j = V_{\mathcal{I}} \begin{bmatrix} D_{j-1} & R_j & \mathbf{0} \end{bmatrix}^T . \tag{2.26}$$

We prove by induction that the user can always reconstruct the secret by iteratively decoding M_i, $i = j, \dots, 1$, in each linear system $V_{\mathcal{I}} M_i$. To that end, we verify the induction hypothesis for $i = j$. Given the system in (2.26), we show that the user can always decode M_j. The zero block matrix in (2.26) is of dimensions $(n - d_j) \times (u\alpha/\alpha_j \alpha_{j-1})$. Therefore, (2.26) can be rewritten as $V'_{\mathcal{I}} \begin{bmatrix} D_{j-1} & R_j \end{bmatrix}^T$, where $V'_{\mathcal{I}}$ is the square Vandermonde matrix of dimensions $d_j \times d_j$ formed by the first d_j columns of $V_{\mathcal{I}}$. Hence, the user can always decode the entries of M_j by inverting $V'_{\mathcal{I}}$.

Next, suppose that the user can decode all the M_i's, $i = j, \dots, l+1$, we prove that the user can always decode M_l. The l^{th} system is given by $V_{\mathcal{I}} M_l$. By construction M_l contains d_l non-zero rows, because the $\mathbf{0}$ block matrix is of dimensions $(n - d_l) \times (u\alpha/\alpha_l \alpha_{l-1})$. In addition, the entries of the last $l - 1$ non-zero rows of M_j are present in D_f for $f = j - 1, \dots, l - 1$, which were previously decoded. It can be checked that $d_j = d_l - (l - 1)$ for all $l < j$. Therefore, after subtracting the last $l - 1$ rows of M_l, the system becomes $V'_{\mathcal{I}} M'_l$, where $V'_{\mathcal{I}}$ is again the $d_j \times d_j$ square Vandermonde matrix defined above and M'_l is the matrix formed of the first $d_j = d_l - (l - 1)$ rows of M_l. Henceforth, the user can always decode M'_l by inverting $V'_{\mathcal{I}}$. Finally, the user can decode all the entires of M_l that consist of the entries of M'_l and the entries of the last $l - 1$ rows of M_l, which were previously decoded.

Next, we show that minimum CO and RO are achieved. The number of symbols read and downloaded by a user contacting d_j parties is equal to $d_j(u\alpha/\alpha_j)$ symbols which corresponds to $d_j u/\alpha_j$ units. Then, the communication and read overheads are given by $d_j u/\alpha_j - u = uz/\alpha_j = uz/(d_j - z)$, which matches the optimal $\mathtt{CO}(d_j)$ and $\mathtt{RO}(d_j)$ for all $d_j = k, \dots, n$, given in (2.4) and (2.5).

3) Perfect privacy: Similarly to the proof of perfect privacy in Theorem 2.1, we need to show that $H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}) = 0$ for all $\mathcal{Z} \subset [n]$, $|\mathcal{Z}| = z$ (see Lemma 2.4). This is equivalent to showing that given the secret $\mathbf{s}$ as side information, any collection $W_{\mathcal{Z}}$ of z shares can decode all the random keys. A collection of $W_{\mathcal{Z}}$ of z shares can be written as $V_{\mathcal{Z}} \begin{bmatrix} M_1 \dots M_h \end{bmatrix}$, which can be divided into $h = n - k + 1$ linear systems as follows,

$$V_{\mathcal{Z}} M_1 = V_{\mathcal{Z}} \begin{bmatrix} S & R_1 \end{bmatrix}^T \tag{2.27}$$

$$V_{\mathcal{Z}} M_2 = V_{\mathcal{Z}} \begin{bmatrix} D_1 & R_2 & \mathbf{0} \end{bmatrix}^T \tag{2.28}$$

$$\vdots$$

$$V_{\mathcal{Z}} M_h = V_{\mathcal{Z}} \begin{bmatrix} D_{h-1} & R_h & \mathbf{0} \end{bmatrix}^T. \tag{2.29}$$

We will prove by induction that given the secret $\mathbf{s}$ as side information, any collection $W_{\mathcal{Z}}$ of z shares can always iteratively decode R_i, $i = 1, \dots, h$, in each linear system $V_{\mathcal{Z}} M_i$. To that end, we verify the induction hypothesis for $i = 1$ by showing that a collection of $W_{\mathcal{Z}}$ shares can always decode R_1 in (2.27). Recall that the dimensions of R_1 are $z \times u\alpha/\alpha_1$. Given the secret $\mathbf{s}$, (2.27) becomes,

$$V_{\mathcal{Z}} \begin{bmatrix} \mathbf{0} & R_1 \end{bmatrix}^T = V''_{\mathcal{Z}} R_1,$$

where $V''_{\mathcal{Z}}$ is a $z \times z$ square Vandermonde matrix formed by the last z columns of $V_{\mathcal{Z}}$. Therefore, R_1 can be decoded by inverting $V''_{\mathcal{Z}}$.

Next, we suppose that any collection of $W_{\mathcal{Z}}$ shares can decode all the R_i's for $i = 1, \dots, l-1$, and show that any collection of $W_{\mathcal{Z}}$ can decode R_l. The l^{th} system is given by $V_{\mathcal{I}} M_l = V_{\mathcal{I}} \begin{bmatrix} D_{l-1} & R_l & \mathbf{0} \end{bmatrix}^T$. By construction, D_{l-1} consists of the entries of the last row of M_{l-1} which were previously decoded. Given the previously decoded information, any collection of

$W_{\mathcal{Z}}$ shares can cancel out the entries of D_{l-1} to obtain $V^*_{\mathcal{Z}} R_l$. Since the dimensions of R_l are $z \times u\alpha/\alpha_l \alpha_{l-1}$, the matrix $V^*_{\mathcal{Z}}$ is a $z \times z$ square Vandermonde matrix formed by the $(\alpha_l + 1)^{\text{st}}$ to $(\alpha_l + z)^{\text{th}}$ rows of $V_{\mathcal{Z}}$. Thus, R_l can be always decoded because $V^*_{\mathcal{Z}}$ is invertible. Therefore, all the keys can always be decoded. Hence, $H(\mathtt{R} \mid \mathtt{W}_{\mathcal{Z}}, \mathtt{S}) = 0$. This concludes the proof of Theorem 2.2.

2.6 Δ-Universal Staircase Codes

We explain how to modify the Universal Staircase codes to construct Staircase codes that achieve minimum CO and RO only for a desired subset Δ of all possible d's, i.e., $\Delta \subseteq \{k, \dots, n\}$. We refer to these codes as (n, k, z, Δ) Δ-universal Staircase codes. The advantage of these codes over universal codes is that they may require smaller number of symbols per share α.

Encoding: Let $\Delta' \triangleq \Delta \setminus \{k\}$ and order the d's in Δ' in decreasing order. We write $\Delta' = \{d_{i_1}, \dots, d_{i_{|\Delta'|}}\} \subseteq \{d_1, \dots, d_{h-1}\}$, where $d_{i_1} > d_{i_2} > \cdots > d_{i_{|\Delta'|}}$. Let $\alpha_{i_j} = d_{i_j} - z$ for all $d_{i_j} \in \Delta'$ and let $\alpha = \text{LCM}(\alpha_1, \dots, \alpha_{|\Delta'|})$. Define $d_{i_{|\Delta'|+1}} \triangleq k$ and $\alpha_{i_{|\Delta'|+1}} \triangleq u$. The secret symbols are arranged in a matrix S of dimensions $\alpha_{d_{i_1}} \times u\alpha/\alpha_{d_{i_1}}$ and the random keys are partitioned into the matrices $R_{i_1}, \dots, R_{i_{|\Delta'|+1}}$, of dimensions $z \times u\alpha/\alpha_{i_1}$ for R_{i_1} and $z \times u\alpha(\alpha_{i_j} - \alpha_{i_{j-1}})/(\alpha_{i_j}\alpha_{i_{j-1}})$ for all other R_{i_j}, $j = 2, \dots, |\Delta'|+1$. Construct M_{i_1} as the $d_{i_1} \times u\alpha/\alpha_{i_1}$ matrix structured as M_1 in (2.9). And, for each d_{i_j}, $j = 2, \dots, |\Delta'| + 1$, construct M_{i_j} as the $d_{i_1} \times u\alpha(\alpha_{i_j} - \alpha_{i_{j-1}})/(\alpha_{i_j}\alpha_{i_j-1})$ structured as M_{i_j} in (2.9). The matrix D_{i_j}, $j = 1, \dots, |\Delta'|$, is the matrix of dimensions $\alpha_{i_{j+1}} \times u\alpha(\alpha_{i_{j+1}} - \alpha_{i_j})/(\alpha_{i_{j+1}}\alpha_{i_j})$ containing the last $d_{i_j} - d_{i_{j+1}}$ rows of $\left[M_{i_1} \dots M_{i_j}\right]$, from row d_{i_j} to row $d_{i_{j+1}} + 1$. Then, concatenate the constructed matrices, $M_{i_1}, \dots, M_{i_{|\Delta'|+1}}$, to obtain the matrix M of dimensions $d_{i_1} \times \alpha$. The matrix M is multiplied by a Vandermonde matrix of dimensions $n \times d_{i_1}$ to obtain the shares.

Decoding: To reconstruct the secret, a user contacting d_{i_j} parties, indexed by $\mathcal{I} \subseteq [n]$, downloads the first $u\alpha/\alpha_{i_j}$ symbols from each contacted party corresponding to $v_i \left[M_{i_1} \dots M_{i_j}\right]$, for all $i \in \mathcal{I}$. The decoding procedure follows the same steps of the decoding procedure presented in the proof of Theorem 2.2 and the user will be able to decode the secret.

We omit the proof of Theorem 2.3 since it follows the same steps of the proof of Theorem 2.2.

2.7 Conclusion

We considered the communication efficient secret sharing (CE-SS) problem. The goal is to minimize the read and download overheads for a user interested in decoding the secret. To that end, we introduced a new class of deterministic linear CE-SS codes, called *Staircase Codes*. We described three explicit constructions of Staircase codes. The first construction achieves minimum overhead for any given number of parties d contacted by the user. The second is a universal construction that achieves minimum overheads simultaneously for all possible values of d. The third construction achieves optimal costs universally for all values of d in a given set $\Delta \subseteq \{k, \ldots, n\}$. All Staircase code constructions can store a secret of maximal size and require a small finite field $\mathbb{F}_q$ of size $q > n$. However, these code constructions require dividing the shares into α symbols.

In conclusion, we point out some problems that remain open. The model we considered here and the proposed Staircase codes can provide security against parties corrupted by a passive Eavesdropper. However, the problem of constructing communication and read efficient codes that provide security against an active (malicious) adversary remains open. Moreover, Staircase codes require dividing the secret into α symbols. No bounds on the minimum value of α is known, but we conjecture that the value of α required by Staircase codes is the minimum one could hope for in order to maintain minimum read overhead.

Chapter 3

Private Coded computation via Staircase Codes

We assume that the workers have similar resources, i.e., similar computation and communication capacities. However, some of the workers may be stragglers, e.g., slow or busy. We are interested in reducing the delays experienced by the Master. We focus on linear computations as an essential operation in many iterative algorithms. We propose a solution based on Staircase codes introduced in the previous chapter. Staircase codes allow flexibility in the number of stragglers up to a given maximum, and universally achieve the information theoretic limit on the download cost by the Master, leading to latency reduction. We find upper and lower bounds on the Master's mean waiting time. We derive the distribution of the Master's waiting time, and its mean, for systems with up to two stragglers. We show that Staircase codes always outperform existing solutions based on classical secret sharing codes. We validate our results with extensive implementation on Amazon EC2.

3.1 Introduction

We consider the setting of distributed computation in which a server `M`, referred to as master, possesses *confidential* data (e.g., personal, genomic or medical data) and wants to perform intensive computations on it. `M` wants to divide these computations into smaller computational tasks and distribute them to n *untrusted* worker machines that can perform these smaller tasks in parallel. The workers then return their results to the master, who can process them to obtain the result of its original task.

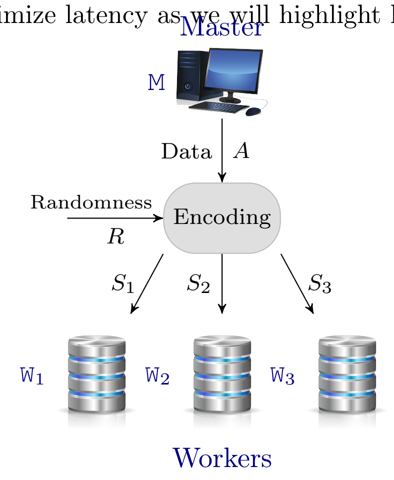
We are interested in applications in which the worker machines do not belong to the same system or cluster as the master. Rather, the workers are online computing machines that can be hired or can volunteer to help the master in its computations, e.g., cloud computing and crowdsourcing platforms like the SETI@home [73] and folding@home [74] projects. The additional constraint, which we worry about here, is that the workers cannot be trusted with the sensitive data, which must remain hidden from them. Privacy could be achieved using fully homomorphic encryption that allows computation over encrypted data. However, homomorphic encryption incurs high computation and storage overheads [75], which may not be feasible in certain applications.

We propose information theoretic privacy to achieve the privacy requirement. Information theoretic privacy is typically used to provide privacy with no constraints on the computational power of the adversary (compromised workers). Our main motivation for information theoretic security is the low complexity of the resulting schemes (compared to homomorphic encryption). The assumption that we have to make here is a limit on the number of workers colluding against the master.

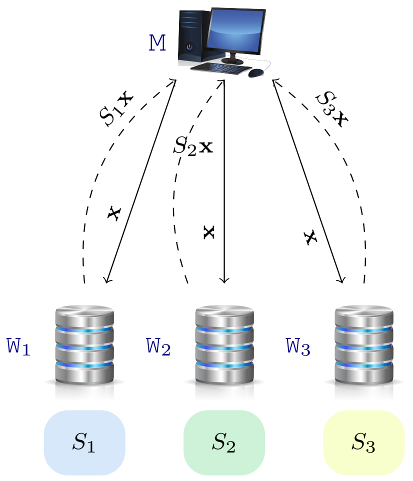
We focus on linear computations (matrix multiplication) since they form a building block of many iterative algorithms, such as principal component analysis, support vector machines and other gradient-descent based algorithms [6, 7]. The workers introduce random delays due to the difference of their workloads or network congestion. This causes the master to wait for the slowest workers, referred to as stragglers in the distributed computation community [5, 76].

Our goal is to reduce the aggregate delay experienced by the master.

Privacy can be achieved by encoding the data, with random keys, using linear secret sharing codes [18] as illustrated in Example 3.1. However, these codes are not specifically designed to minimize latency as we will highlight later.



(a) The master M encodes its data A with a random matrix R into 3 secret shares S_1, S_2, S_3. Any two shares can decode A. For example, $S_1 = R$, $S_2 = A + R$, and $S_3 = A + 2R$. M sends the share S_i to worker W$_i$. The randomness R is used to ensure privacy.



(b) To compute $A\mathbf{x}$, M sends $\mathbf{x}$ to all the workers. Each worker W$_i$ computes $S_i\mathbf{x}$ and sends the result to M. M can decode $A\mathbf{x}$ after receiving any two responses, e.g., $A\mathbf{x} = S_2\mathbf{x} - S_1\mathbf{x} = (A + R)\mathbf{x} - R\mathbf{x}$.

Figure 3.1: Private distributed matrix multiplication with 3 workers. The master encodes its data using a linear secret sharing code, e.g., Shamir's codes (given in the caption) [8, 53] or Staircase codes (given in Table 3.1) [24, 25]. Decoding $A\mathbf{x}$ follows from the linearity of the code.

Example 3.1. *Let the matrix A denote the data set owned by the master and let $\mathbf{x}$ be a given vector.* M *wants to compute $A\mathbf{x}$. Suppose that* M *gets the help of* 3 *workers out of which at most* 1 *may be a straggler.* M *generates a random matrix R of same dimensions as A with entries drawn from the same alphabet as the entries of A.* M *encodes A and R into 3 shares $S_1 = R$, $S_2 = R + A$ and $S_3 = R + 2A$ using a secret sharing scheme [8, 53].* M *sends share S_i to worker* W$_i$ *(Figure 3.1a) and then sends $\mathbf{x}$ to all the workers. Each worker computes $S_i\mathbf{x}$ and sends it back to* M *(Figure 3.1b).* M *can decode $A\mathbf{x}$ after receiving any* 2 *responses. For instance, if the first two workers respond,* M *can obtain $A\mathbf{x} = S_2\mathbf{x} - S_1\mathbf{x}$. No information about A is revealed to the workers, because A is one-time padded by R.*

In the previous example, even if there were no stragglers, M still has to wait for the full responses of two workers, and the response of the third one will not be used for decoding. In

S_1	S_2	S_3
$A_1 + A_2 + R_1$	$A_1 + 2A_2 + 4R_1$	$A_1 + 3A_2 + 4R_1$
$R_1 + R_2$	$R_1 + 2R_2$	$R_1 + 3R_2$

Table 3.1: The shares sent by M to each worker when using Staircase codes. In this example, each share is divided into two sub-shares. The master sends $\mathbf{x}$ to all the workers. Each worker multiplies the sub-shares by $\mathbf{x}$ (going from top to bottom) and sends each multiplication back to M independently. The master can decode $A\mathbf{x}$ by receiving the multiplication of $\mathbf{x}$ by either the first sub-share of each worker (in black) or two sub-shares from any two workers (in black and blue). After receiving enough sub-shares, the master sends a stop message to the workers telling them to stop computing on the remaining sub-shares. Note that if M uses the first three sub-shares, it only decodes half of $R\mathbf{x}$, i.e., $R_1\mathbf{x}$, and does not need to decode $R_2\mathbf{x}$. The operations shown are in $\mathbb{F}_5$.

addition, M always has to decode $R\mathbf{x}$ in order to decode $A\mathbf{x}$. Hence, more delays are incurred by spending communication and computation resources on decoding $R\mathbf{x}$, which is only needed for privacy. We overcome those limitations by using Staircase codes introduced in [24, 25] (see Chapter 2 for more details) which do not always require decoding $R\mathbf{x}$. Thus, possibly reducing the computation load at the workers and the communication cost at the master. In addition, Staircase codes allow more flexibility in the number of responses needed for decoding $A\mathbf{x}$, as explained in the next example.

Example 3.2 (Staircase codes)**.** *Consider the same setting as Example 3.1. Instead of using a classical secret sharing code,* M *now encodes A and R using the Staircase code given in Table 3.1. The Staircase code requires* M *to divide the matrices A and R into $A = \begin{bmatrix} A_1^T & A_2^T \end{bmatrix}^T$ and $R = \begin{bmatrix} R_1^T & R_2^T \end{bmatrix}^T$. In this setting,* M *sends two sub-shares to each worker, hence each task consists of 2 sub-tasks. The master sends $\mathbf{x}$ to all the workers. Each worker multiplies the sub-shares by $\mathbf{x}$ (going from top to bottom) and sends each multiplication back to* M *independently. Now,* M *has two possibilities for decoding:* 1) M *receives the first sub-task from all the workers, i.e., receives $(A_1 + A_2 + R_1)\mathbf{x}$, $(A_1 + 2A_2 + 4R_1)\mathbf{x}$ and $(A_1 + 2A_2 + 4R_1)\mathbf{x}$ and decodes $A\mathbf{x}$ which is the concatenation of $A_1\mathbf{x}$ and $A_2\mathbf{x}$. Note that here* M *decodes only $R_1\mathbf{x}$ and does not need to decode $R_2\mathbf{x}$.* 2) M *receives all the sub-tasks from any* 2 *workers and decodes $A\mathbf{x}$. Here* M *has to decode $R_1\mathbf{x}$ and $R_2\mathbf{x}$. After receiving enough sub-tasks, the master sends a message to the workers instructing them to stop computing the remaining sub-tasks. One can check that no information about A is revealed to the workers, because each sub-share is padded by a random* *matrix.*

Under a shifted exponential delay model for each worker, we show that the Staircase code given in Example 3.2 can lead to a 45% improvement in delay over the secret sharing code given in Example 3.1. Our goal is to give a general systematic study of the delay incurred by Staircase codes and compare it to classical secret sharing codes.

3.1.1 Contributions

To the extent of our knowledge, this is the first work to analyze latency for private distributed coded computation under the presence of stragglers. We consider the distributed computation setting described above in which we require the workers to learn no information (in an information theoretic sense) about the master's data. We study the waiting time of the master caused by delays of the workers. We follow the literature, e.g., [6, 26], and model the service time at the workers as a shifted exponential random variable. This service time includes upload time, computation time and download time, i.e., computation and network latency. Finding codes that minimize the delay at the master is still an open problem in general. In this work, we take the download communication cost as a proxy for delay when designing the coding schemes. More precisely, we study the performance of Staircase codes [24,25] that achieve the information theoretic limit on download cost [23] and compare them to classical secret sharing codes.

Before we state our contributions, we introduce some necessary notations. We denote by n the number of workers available to help the master, by k the minimum number of non stragglers and by z the maximum number of colluding workers. We refer to such private distributed computation system by an (n, k, z) system. Our contributions are summarized by the following:

1. *General bounds for systems with any number of stragglers:* We derive an upper and a lower bound on the master's mean waiting time when using Staircase codes (Theorem 3.1). Moreover, we derive the exact distribution of the master's waiting time when using Staircase codes, in an integral form (Theorem 3.4). Using the upper bound, we compare the performance of Staircase codes to classical secret sharing codes and characterize the

savings obtained by Staircase codes. We show that Staircase codes always outperform classical secret sharing codes.

2. *Exact characterization for systems with up to* 2 *stragglers:* We use the integral expression of Theorem 3.4 to find the exact distribution of the master's waiting time for systems with up to $n-k=1$ and up to $n-k=2$ stragglers (Corollary 3.5). Moreover, we derive the exact expressions of the master's mean waiting time for these systems (Theorem 3.2) and use these expressions to show the tightness of our upper bound.

3. *Simulations and validation:* We ran extensive MATLAB simulations for different system parameters. We focus on two regimes: regime with fixed rate k/n and regime with fixed maximum number of stragglers $n-k$. Our main observation is that the upper bound, based on Jensen's inequality, is a good approximation of the mean waiting time. Furthermore, we validate our results with extensive implementation on Amazon EC2 clusters. The savings obtained on EC2 clusters are within the range of the values predicted by the theoretical model. To give an example, for $n = 4$ workers, large data and high traffic regime, our implementation shows 59% (Figure 3.8a) savings in the mean waiting time while the theoretical model predicts 66% savings (Figure 3.5a).

3.1.2 Organization

The chapter is organized as follows. We formalize the problem and define the model in section 3.2. In section 3.3, we present and discuss our main results. In Sections 3.4 and 3.5, we study the probability distribution of the master's waiting time and derive bounds on the mean waiting time. We show, in Section 3.6, that the (random) number of workers that minimizes the waiting time is concentrated around its average. We evaluate the performance of Staircase codes via simulation in Section 3.7. In Section 3.8, we give a representative sample of our implementation on Amazon EC2 clusters and compare them to our theoretical findings. We give an overview of the related works in Section 3.9. We conclude the chapter in section 4.7. We prove Theorem 3.2 and discuss how our scheme can be extended to provide privacy over multiple iterations of an algorithm in Appendix B.1 and Appendix A, respectively.

3.2 System Model

We consider a master server $\mathtt{M}$ which wants to perform intensive computations on confidential data represented by an $m \times \ell$ matrix A (typically $m >> \ell$). In machine learning applications m denotes the number of data points (examples) possessed by $\mathtt{M}$ and ℓ denotes the number of attributes (features) of each example. $\mathtt{M}$ divides these computations into smaller computational tasks and assigns them to n workers $\mathtt{W}_i$, $i = 1, \ldots, n$, that can perform these tasks in parallel. The division is horizontal, i.e., each worker gets a given number of rows of A with all their corresponding columns.

3.2.1 Computations model

We focus on linear computations. The motivation is that a building block in several iterative machine learning algorithms, such as gradient descent, is the multiplication of A by a sequence of $\ell \times 1$ attribute vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots$. In the sequel, we focus on the multiplication $A\mathbf{x}$ with one attribute vector $\mathbf{x}$.

3.2.2 Workers model

The workers have the following properties: *1)* The workers incur random delays while executing the task assigned to them by $\mathtt{M}$ resulting in what is known as the straggler problem [5,6,76]. We model all the delays (including the computation and communication delay) incurred by each worker by an independent and identical shifted exponential random variable. This distribution captures the constant part of the task-dependent service time at each server, along with the unpredictable randomness in service due to uncorrelated background processes at each server. Further, this distribution is widely used to model service in compute cluster [6, 26], and lends itself well to analytical tractability. *2)* Up to z, $z < k$, workers can collude, i.e., at most z workers can share with each other the data they receive from $\mathtt{M}$. The threshold z could be thought of as a desired level of security. This has implications on the privacy constraint described later.

3.2.3 General scheme

$\mathtt{M}$ encodes A, using randomness, into n shares S_i sent to worker $\mathtt{W}_i$, $i = 1, \dots, n$. Any k or more shares can decode A, and any collection of z workers obtain zero information about A. For any set $\mathcal{B} \subseteq \{1, \dots, n\}$, let $S_{\mathcal{B}} = \{S_i, i \in \mathcal{B}\}$ denote the collection of shares given to worker $\mathtt{W}_i$ for all $i \in \mathcal{B}$. Let $\mathtt{A}$ denote the random variable representing A and $\mathtt{S}_{\mathcal{B}}$ denote the collection of random variable representing the shares indexed by $\mathcal{B}$. The previous requirements can be expressed as,

$$H(\mathtt{A}|\mathtt{S}_{\mathcal{B}}) = 0, \quad \forall \mathcal{B} \subseteq \{1, \dots, n\} \text{ s.t. } |\mathcal{B}| \geq k,$$

$$H(\mathtt{A}|\mathtt{S}_{\mathcal{Z}}) = H(\mathtt{A}), \quad \forall \mathcal{Z} \subseteq \{1, \dots, n\} \text{ s.t. } |\mathcal{Z}| \leq z.$$

At each iteration, the master sends $\mathbf{x}$ to all the workers. Then, each worker computes $S_i\mathbf{x}$ and sends it back to the master. In the case where the share S_i consists of sub-shares, each worker multiplies the sub-shares by $\mathbf{x}$ and sends the result back to the master independently. Since the scheme and the computations are linear, the master can decode $A\mathbf{x}$ after receiving enough responses. After receiving enough responses the master sends a stop message to the workers instructing them to stop computing on the remaining sub-shares. We refer to such scheme as an (n, k, z) system. We note that our scheme can be generalized to the cases the where the attribute vectors $\mathbf{x}$ contain information about A, and therefore need to be hidden from the workers. We describe the generalization of our scheme to such case in Appendix A.

3.2.4 Encoding

We consider classical secret sharing codes [8, 53] and universal Staircase codes [24, 25]. We now describe their properties that are necessary for the delay analysis (detailed description can be found in Section 2.3.2). Secret sharing codes require the division of A into $k - z$ row blocks each of dimension $\frac{m}{k-z} \times \ell$ and encodes them into n shares of the identical dimension. Any k shares can decode A. Similarly, Staircase codes encode A into n shares of $\frac{m}{(k-z)} \times \ell$ each with the additional requirement that each share is divided into $\alpha = \text{LCM}\{k-z+1, \dots, n-z\}$ sub-shares. Decoding A requires $\delta_d \alpha$ sub-shares, $\delta_d \triangleq \frac{(k-z)}{(d-z)}$, from any of the d shares, $d \in \{k, \dots, n\}$. We show that Staircase codes outperform classical codes in terms of incurred delays.

3.2.5 Delay model

Let T_A be the random variable representing the time spent to compute $A\mathbf{x}$ at one worker. We assume a mother runtime distribution $F_{T_A}(t)$ that is shifted exponential with rate λ and a constant shift c. As mentioned before, this is distribution is widely used for modeling service time in a compute cluster [6,26], and primarily motivated by its analytical tractability. Furthermore, the shifted exponential distribution captures the two parts of the service completion time: the constant part of the task-dependent service time at each server, and the stochasticity in service due to uncorrelated background processes at each server. For each $i \in \{1, \dots, n\}$, we let T_i denote the time spent by worker $\mathtt{W}_i$ to execute its task. Due to the encoding, each task given to a worker is $k-z$ times smaller than A, or $T_i = \frac{T_A}{(k-z)}$. It follows that F_{T_i} is a scaled distribution of F_{T_A}. That is,

$$F_{T_i}(t) \triangleq F_{T_A}((k-z)t) = 1 - e^{-\lambda(k-z)(t-\frac{c}{k-z})}, \quad \text{for } t \geq c/(k-z). \tag{3.1}$$

For an (n, k, z) system using Staircase codes, we assume that T_i is evenly distributed among the sub-tasks[1]. That is, the time spent by a worker $\mathtt{W}_i$ on one sub-task is equal to T_i/α, and the time spent on $\alpha\delta_d = \alpha\frac{k-z}{d-z}$ sub-tasks is $\delta_d T_i$.

Let $T_{(i)}$ be the i^{th} order statistic of the T_i's and $T_{\text{SC}}(n, k, z)$ be the time the master waits until it can decode $A\mathbf{x}$. If the aggregate wait is due to d workers each finishing δ_d fraction of

[1]Therefore, we make two assumptions on the waiting time of the sub-tasks: (1) the parameters of its distribution (effective c and λ) vary linearly with the sub-task size and (2) the waiting time of sub-tasks of the same task take equal service time, and therefore are not independent. These assumptions make the problem more amenable to theoretical analysis. In Section 3.8, we compare our model to traces obtained from Amazon cloud and show that our model provides insightful engineering guidelines.

its α sub-tasks, then the master's waiting time is $\delta_d T_{(d)}$. We can write

$$T_{\text{SC}}(n,k,z) = \min_{d \in \{k,\dots,n\}} \left\{ \delta_d T_{(d)} \right\}. \tag{3.2}$$

It is useful for our analysis to look at T_i as the sum of an exponential random variable T_i' and a constant offset, i.e.

$$T_i = T_i' + c/(k-z), \text{ where } T_i' \sim \exp\left(\lambda(k-z)\right).$$

From this interpretation, it is easy to verify that the d^{th} order statistic $T_{(d)}$ of $(T_1, T_2, \dots, T_n)$ can be expressed as

$$T_{(d)} = T_{(d)}' + c/(k-z),$$

where $T_{(d)}'$ is the d^{th} order statistic of n *iid* exponential random variables with rate $\lambda(k-z)$. Therefore, we can write the master's waiting time for Staircase codes as

$$T_{\text{SC}}(n,k,z) = \min_{d \in \{k,\dots,n\}} \left\{ \delta_d \Big(T_{(d)}' + \frac{c}{k-z} \Big) \right\}. \tag{3.3}$$

For an (n,k,z) system using classical secret sharing codes, the master's waiting time $T_{\text{SS}}(n,k,z)$ is equal to the time spent by the fastest k workers to finish their individual tasks. Hence, we can write

$$T_{\text{SS}}(n,k,z) = T_{(k)}. \tag{3.4}$$

We drop the (n,k,z) notation from $T_{\text{SC}}(n,k,z)$ and $T_{\text{SS}}(n,k,z)$ when the system parameters are clear from the context.

3.3 Main Results

We characterize the delay performance of private coded computation when using Staircase codes and compare it to classical secret sharing codes. The performance of Staircase codes is reflected

Symbol	Meaning	Symbol	Meaning
n	number of workers	$\mathtt{M}$	master server
k	threshold on non stragglers	$\mathtt{W}_i$	worker i
z	number of colluding workers	A	data matrix of the master
d	number of non stragglers	$\mathbf{x}$	attribute vector of dimension $\ell \times 1$
m	number of rows in A	S_i	share sent to worker i
ℓ	number of columns in A	T_A	time to compute $A\mathbf{x}$ at one worker
α	number of sub-shares in one share	T_i	time spent to compute one task at the worker
c	shift of the shifted exponential distribution	$T_{(i)}$	i^{th} order statistic of $T_1, \dots, T_n$
λ	rate of the shifted exponential distribution	T_{SC}	master's waiting time when using Staircase codes
δ_d	fraction of the share needed when d workers are not stragglers	T_{SS}	master's waiting time when using secret sharing
H_n	Harmonic number $H_h \triangleq \sum_{i=1}^{n} 1/i$	T'_A	exponential random variable, $T'_A = T_A + c$

Table 3.2: Summary of notations for this Chapter.

in the master's waiting time T_{SC}. Towards our goal, we establish in Theorem 3.1 general bounds on the master's mean waiting time $\mathbb{E}[T_{\text{SC}}(n,k,z)]$ when using Staircase codes for all (n,k,z) systems, under the shifted exponential delay model.

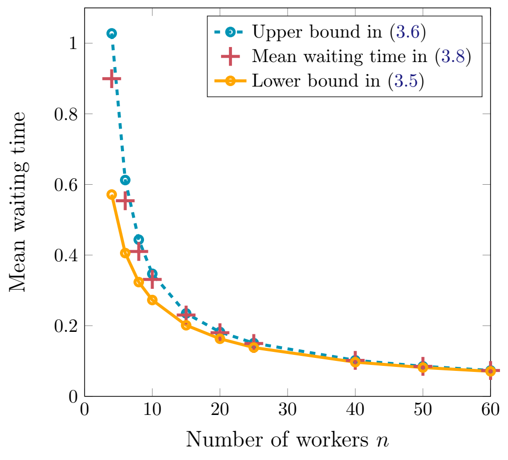
Theorem 3.1 (Bounds on the master's mean waiting time $\mathbb{E}[T_{\text{SC}}]$)**.** *Let H_n be the n^{th} harmonic sum defined as $H_n \triangleq \sum_{i=1}^{n} \frac{1}{i}$, with the notation $H_0 \triangleq 0$. The mean waiting time of the master $\mathbb{E}[T_{SC}]$ for an (n,k,z) Staircase coded system is upper bounded by*

$$\mathbb{E}[T_{SC}] \leq \min_{d\in\{k,\dots,n\}} \left(\frac{H_n - H_{n-d}}{\lambda(d-z)} + \frac{c}{d-z} \right), \tag{3.5}$$

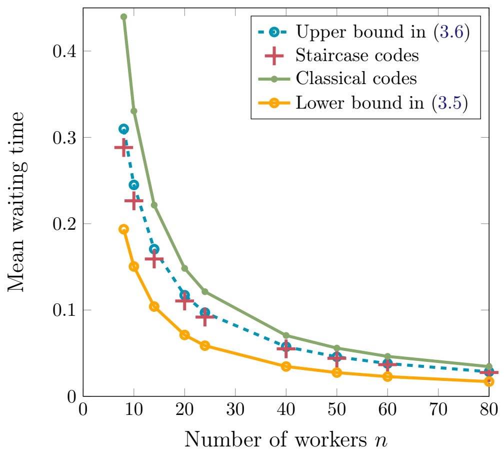
and lower bounded by

$$\mathbb{E}[T_{SC}] \geq \frac{c}{n-z} + \max_{d\in\{k,\dots,n\}} \sum_{i=0}^{k-1} \binom{n}{i} \sum_{j=0}^{i} \binom{i}{j} \frac{2(-1)^j}{\lambda\left(2(n-i+j)(d-z) + (n-d)(n-d+1)\right)}. \tag{3.6}$$

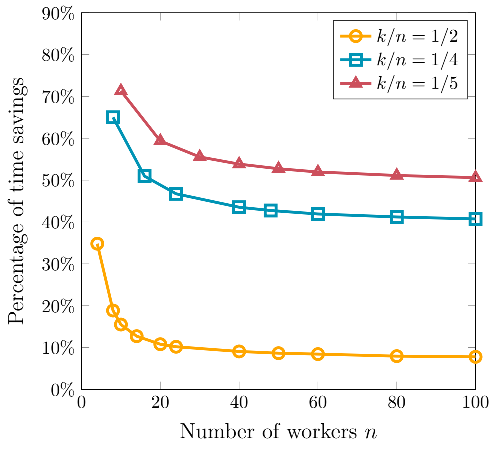
We derive in Section 3.5 a general integral expression (B.3) leading to the CDF $F_{T_{\text{SC}}}(t)$ of T_{SC}, the waiting time of the master for all (n,k,z) systems. Using the general integral expression, we derive the exact expression of the CDF $F_{T_{\text{SC}}}(t)$ for systems with $n = k+1$ and $n = k+2$ as stated in the next Theorem.



(a) Systems with $n - k = 2$ stragglers.



(b) Systems with fixed rate $k/n = 1/2$.



(c) Savings for systems with fixed k/n.

Figure 3.2: Theoretical upper and lower bounds for systems with rate of the exponential random variable $\lambda = 1$, shift $c = 1$ and no colluding workers, i.e., $z = 1$. Figure 4.3a compares the bounds derived in Theorem 3.1 to the theoretical mean waiting time for $(k+2, k, 1)$ derived in Corollary 3.2. Observe that the upper bound in (3.5) is a good approximation of the mean waiting time in (3.8). Figure 4.3b compares the bounds in (3.5) and (3.6) to the simulated mean waiting time for (n, k, z) systems with fixed rate $k/n = 1/2$. We obtain the mean waiting time by averaging over 10000 iterations for each value of n. Figure 4.2b compares the upper bound in (3.5) to the mean waiting time of classical secret sharing in (3.9). The savings are computed as the normalized difference between the waiting time of Staircase codes and classical secret sharing codes, i.e., $\left(\mathbb{E}[T_{\text{SC}}] - \mathbb{E}[T_{\text{SS}}]\right)/\mathbb{E}[T_{\text{SS}}]$.

Theorem 3.2 (Exact expression of $\mathbb{E}[T_{\text{SC}}]$ for systems with up to 2 stragglers)**.** *The mean waiting time of the master for* $(k+1,k,z)$ *and* $(k+2,k,z)$ *systems is given in* (3.7) *and* (3.8)*, respectively.*

$$\mathbb{E}\left[T_{SC}(k+1,k,z)\right] = \frac{c}{k-z+1} + \frac{1}{\lambda}\sum_{i=1}^{k+1}(-1)^i\binom{k+1}{i}\left[\frac{i\exp\left(\frac{-\lambda c}{k-z}\right)}{(k-z)i+1} - \frac{1}{(k-z+1)i}\right]. \tag{3.7}$$

$$\begin{aligned}\mathbb{E}[T_{SC}(k+2,k,z)] &= \mathbb{E}[T_{SC}(k+2,k+1,z)] \\ &\quad + \frac{1}{\lambda}\sum_{i=2}^{k+2}(-1)^i\binom{k+2}{i}\binom{i}{2}\left[\frac{\exp\left(-\frac{4\lambda c}{k-z}\right)}{(k-z)i+4} - \frac{2\exp\left(-\frac{3\lambda c}{k-z}\right)}{(k-z)i+3}\right].\end{aligned} \tag{3.8}$$

To give insights into the theoretical bounds above, we compare in Figure 4.3a bounds (3.5) and (3.6) for the case of $n = k+2$ to the exact expression in (3.8). We see that the upper bound in (3.5) is closer to the actual value and the gap between the two bounds closes as n increases. We also establish the comparison for fixed rate regimes, in particular rate $k/n = 1/2$. Since here $n \geq k+2$, we compare in Figure 4.3b the bounds to numerical results obtained by simulation and observe the same behavior as before. We also plot in the same figure the mean waiting time for classical secret sharing codes obtained from (3.4) and given by

$$\mathbb{E}[T_{\text{SS}}] = \frac{H_n - H_{n-k}}{\lambda(k-z)} + \frac{c}{k-z}. \tag{3.9}$$

This allows to verify that Staircase codes always outperform classical secret sharing codes. In Figure 4.2b, we plot the lower bound on the relative savings brought by Staircase codes for systems with rate $k/n = 1/2, 1/4, 1/5$. For instance, for rate $1/4$, the savings are lower bounded by 40% for large n. We supplement our theoretical results in Section 3.7 with an extensive array of simulations in addition to measurement results obtained by implementation on Amazon EC2 clusters. The savings obtained in the implementation on Amazon cloud are within the savings predicted by the theoretical model.

3.4 Bounds on the Master's Mean Waiting Time for all (n, k, z) Systems

We derive an upper and a lower bound on the master's mean waiting time $\mathbb{E}[T_{\text{SC}}(n,k,z)]$ for all (n,k,z) systems, i.e., we prove Theorem 3.1. We divide the proof into two parts: proof of the upper bound, and proof of the lower bound.

3.4.1 Proof of the upper bound on the mean waiting time

Proof. We use Jensen's inequality to upper bound the mean waiting time $\mathbb{E}[T_{\text{SC}}]$. Since min is a convex function, we can use Jensen's inequality to upper bound the mean waiting time,

$$\mathbb{E}[T_{\text{SC}}] = \mathbb{E}\left[\min_{d\in\{k,\dots,n\}}\left\{\delta_d T'_{(d)} + \frac{c}{d-z}\right\}\right] \leq \min_{d\in\{k,\dots,n\}}\left\{\delta_d \mathbb{E}\left[T'_{(d)}\right] + \frac{c}{d-z}\right\}. \tag{3.10}$$

We need the following Theorem in order to derive an exact expression of the mean of the d^{th} order statistic of n *iid* exponential random variables.

Theorem (Renyi [77])**.** *The d^{th} order statistic $T'_{(d)}$ of n* iid *exponential random variables T'_i is equal to the following random variable in the distribution*

$$T'_{(d)} \triangleq \sum_{j=1}^{d} \frac{T'_j}{n-j+1}.$$

Using Renyi's Theorem, the mean of the d^{th} order statistic $\mathbb{E}\left[T'_{(d)}\right]$ can be written as

$$\mathbb{E}[T'_{(d)}] = \mathbb{E}[T'_j]\sum_{j=0}^{d-1}\frac{1}{n-j} = \frac{H_n - H_{n-d}}{\lambda(k-z)}. \tag{3.11}$$

From equations (3.10) and (3.11), the mean waiting time is upper bounded by

$$\mathbb{E}[T_{\text{SC}}] \leq \min_{d\in\{k,\dots,n\}}\left\{\frac{H_n - H_{n-d}}{\lambda(d-z)} + \frac{c}{d-z}\right\}.$$

We give an intuitive behavior of the upper bound. The harmonic number can be approximated by $H_n \approx \log(n)+\gamma$, where $\gamma \approx 0.577218$ is called the Euler-Mascheroni constant. Alternatively, we can use the upper and lower bounds $\log(n) < H_n < \log(n+1)$ on the Harmonic number H_n, to upper bound the mean waiting time

$$\mathbb{E}[T_{\mathrm{SC}}] < \min\left\{\min_{d\in\{k,\dots,n-1\}}\left\{\frac{1}{\lambda(d-z)}\log\left(\frac{n+1}{n-d}\right)+\frac{c}{d-z}\right\}, \frac{1}{\lambda(n-z)}\log(n+1)+\frac{c}{n-z}\right\}.$$

□

3.4.2 Proof of the lower bound on the mean waiting time

Proof. Recall that $T_{SC} = \min\{\delta_d T_{(d)} : d \in \{k,\dots,n\}\} = \min\{\delta_d T'_{(d)} + \frac{c}{d-z} : d \in \{k,\dots,n\}\}$. Since the minimum of the sum is greater than the sum of the minimums, we can lower bound the waiting time T_{SC} in terms of residual waiting time $T'_{\mathrm{SC}} \triangleq \min\{\delta_d T'_{(d)} : d \in \{k,\dots,n\}\}$, as

$$T_{\mathrm{SC}} = \min_{d\in\{k,\dots,n\}}\{\delta_d T'_{(d)} + \frac{c}{d-z}\} \geq T'_{\mathrm{SC}} + \frac{c}{(n-z)}.$$

Since the mean of a continuous random variable can be computed by integrating the tail probability, we lower bound $\mathbb{E}[T'_{\mathrm{SC}}]$ by lower bounding the tail probability of T'_{SC} exceeding any threshold value t. We observe that T'_{SC} is greater than t, if and only if the d^{th} order statistic $T'_{(d)}$'s is greater than $\frac{t}{\delta_d}$ for each $d \in \{k,\dots,n\}$. That is,

$$\{T'_{\mathrm{SC}} > t\} = \bigcap_{d=k}^{n}\left\{T'_{(d)} > \frac{t}{\delta_d}\right\}.$$

Recall that $t\delta_d^{-1}(k-z) = t(d-z)$ is increasing in d, and so is $T'_{(d)}$. For the residual service times $T'_1,\dots,T'_n$, we consider the following set

$$\mathcal{C}_d(t) \triangleq \left\{T'_{(k)} > \frac{t}{\delta_d}\right\}\bigcap_{i=d+1}^{n}\left\{T'_{(i)} - T'_{(i-1)} > \frac{t}{\delta_i} - \frac{t}{\delta_{i-1}}\right\}.$$

For each $d \in \{k, \dots, n\}$, we observe that $\mathcal{C}_d(t) \subseteq \{T'_{\text{SC}} > t\}$ since $\{T'_{(k)} > t\delta_d^{-1}\} \subseteq \cap_{j=k}^{d} \{T'_{(j)} > t\delta_j^{-1}\}$. It follows that, $\Pr\{T'_{\text{SC}} > t\} \geq \max_{d \in \{k,\dots,n\}} \Pr(\mathcal{C}_d(t))$. Next, we evaluate $\Pr(\mathcal{C}_d(t))$ explicitly. To this end, we first observe that ${\delta_j}^{-1} - {\delta_{j-1}}^{-1} = (k-z)^{-1}$ identically for each $j \in \{1, \dots, n\}$. Further, we apply Renyi's Theorem and independence of residual times T'_is to write

$$\Pr\left(\mathcal{C}_d(t)\right) = \Pr\left\{T'_{(k)} > \frac{t}{\delta_d}\right\} \prod_{j=d+1}^{n} \Pr\left\{\frac{T'_j}{n-j+1} > \frac{t}{(k-z)}\right\}. \tag{3.12}$$

In the following, we would use $F(t) = 1 - e^{-\lambda t}$ for $t \geq 0$ to represent the cumulative distribution function (CDF) and $\bar{F}(t) = 1 - F(t)$ to represent the complementary cumulative distribution function (CCDF), of an exponential random variable with rate λ. It follows that the CCDF for the residual service time T'_i is $\Pr\{T'_j > t\} = \bar{F}((k-z)t)$. Utilizing the exponential form, we can write

$$\prod_{j=d+1}^{n} \Pr\left\{\frac{T'_j}{n-j+1} > \frac{t}{(k-z)}\right\} = \bar{F}\left(\sum_{j=d+1}^{n} (n-j+1)t\right) = \bar{F}\left(\frac{(n-d)(n-d+1)t}{2}\right). \tag{3.13}$$

From definition, it follows that $\delta_k = 1$. Further, the kth order statistic of n residual service times exceeds a threshold if and only if at most $k-1$ different residual service times are less than the threshold, c.f., Lemma 3.3. That is,

$$\Pr\left\{T'_{(k)} > t\right\} = \sum_{i=0}^{k-1} \binom{n}{i} F\left((k-z)t\right)^i \bar{F}\left((k-z)t\right)^{n-i}. \tag{3.14}$$

Since $F(t) = 1 - \bar{F}(t)$, using the binomial expansion, we have

$$F\left((k-z)t\right)^i = \sum_{j=0}^{i} \binom{i}{j} (-1)^j \bar{F}\left((k-z)t\right)^j. \tag{3.15}$$

Exploiting the exponential form of $\bar{F}(t)$, aggregating results from (3.13), (3.14) and (3.15), we can re-write (3.12) as

$$\Pr\left(\mathcal{C}_d(t)\right) = \sum_{i=0}^{k-1}\binom{n}{i}\sum_{j=0}^{i}\binom{i}{j}(-1)^j \bar{F}\big(t(n-i+j)(d-z) + t(n-d)(n-d+1)/2\big). \tag{3.16}$$

The proof follows from the integral $\int_0^\infty e^{-xt}dt = \frac{1}{x}$, the linearity of integrals, and the following lower bound

$$\mathbb{E}[T'_{\text{SC}}] = \int_0^\infty \Pr\left\{T'_{\text{SC}} > t\right\} dt \geq \int_0^\infty \max_{d\in\{k,\ldots,n\}} \Pr(\mathcal{C}_d(t))dt \geq \max_{d\in\{k,\ldots,n\}} \int_0^\infty \Pr(\mathcal{C}_d(t))dt.$$

□

Lemma 3.3. *Marginal complementary distribution of d^{th} order statistics $T'_{(d)}$ of n* iid *random variables $(T'_1,\ldots,T'_n)$ with common distribution $f_{T'}(t)$ is given by*

$$\Pr\{T'_{(d)} > t\} = \sum_{i=0}^{d-1}\binom{n}{i} F_{T'}(t)\bar{F}_{T'}(t)^{n-i}.$$

We note the cumulative distribution function (CDF) of f by $F_{T'}(t) \triangleq f_{T'}(T' < t)$ and the complementary cumulative distribution function (CCDF) of f by $\bar{F} \triangleq f_{T'}(T' > t) = 1 - F_{T'}(t)$.

Proof. The d^{th} order statistic is greater than t, if and only if at most $d-1$ out of n *iid* random variables $(T_1,\ldots,T_n)$ can be less than t, and the rest are greater than t. □

3.5 Distribution of the Master's Waiting Time for all (n,k,z) Systems

Now we are ready to derive an integral expression for the probability distribution of T_{SC}, the master's waiting time when using Staircase codes.

Theorem 3.4 (Integral expression leading to $F_{T_{\text{SC}}}(t)$). *The distribution of the master's waiting* *time T_{SC} of an (n, k, z) system using Staircase codes is given by*

$$F_{T_{SC}}(t) = 1 - n! \int_{(y_k,\dots,y_n)\in\mathcal{A}(t)} \frac{F_{T'}(y_k)^{k-1}}{(k-1)!} dF_{T'}(y_k)\dots dF_{T'}(y_n) \quad \text{for } t > 0. \tag{3.17}$$

We denote the residual service time at each worker $\mathtt{W}_i$, $i = 1, \dots, n$, by the random variable $T_i' = T_i - \frac{c}{k-z}$, and the associated distribution by $F(y_i) \triangleq F_{T'}(y_i) = 1 - \exp(-\lambda y_i)$ for $y_i > 0$. For $i = k, \dots, n$, we define t_i as $t_i \triangleq \max\left\{\left(\frac{i-z}{k-z}\right)\left(t - \frac{c}{i-z}\right), 0\right\}$. We denote by $\mathcal{A}(t)$ the set of ordered variables $(y_k, \dots, y_n)$ such that

$$\mathcal{A}(t) \triangleq \{0 \le y_k \le y_{k+1} \le \dots \le y_n : t_k < y_k, \dots, t_n < y_n\}.$$

We apply Theorem 3.4 to get the mean waiting time of the master and the exact distribution of the waiting time for systems with $n = k+1$ and $n = k+2$ in Theorem 3.2 and Corollary 3.5, respectively.

Corollary 3.5 (Exact expression of $F_{T_{\text{SC}}}(t)$ for systems with up to 2 stragglers)**.** *The distribution of the master's waiting time for $(k+1, k, z)$ and $(k+2, k, z)$ systems is given in (3.18) and (3.19), respectively.*

$$F_{T_{SC(k+1,k,z)}}(t) = F_{T'}(t_{k+1})^{k+1} + F_{T'}(t_k)^k \bar{F}_{T'}(t_{k+1})(k+1). \tag{3.18}$$

$$\begin{aligned} F_{T_{SC(k+2,k,z)}}(t) = F_{T'}(t_{k+2})^{k+2} + (k+2)\bar{F}_{T'}(t_{k+2})\Big[F_{T'}(t_{k+1})^{k+1} \\ + (k+1)F_{T'}(t_k)^k(\bar{F}_{T'}(t_{k+1}) - \frac{1}{2}\bar{F}_{T'}(t_{k+2}))\Big]. \end{aligned} \tag{3.19}$$

Both distributions are defined for $t > 0$, and $F_{T'}(t) \triangleq 1 - \exp(-\lambda(k-z)t)$.

We omit the proof of Corollary 3.5 since it follows from simply integrating (B.3) and defer the proofs of Theorem 3.2 and Theorem 3.4 to Appendix B.1 and Appendix B.2, respectively.

3.6 Interplay Between Code Design and Latency

We have seen so far that universal Staircase codes allows the master to decode $A\mathbf{x}$ from any random number d of workers, $k \leq d \leq n$. However, the downside is that the universal construction requires a large number of sub-tasks $\alpha = \text{LCM}\{k-z+1, \dots, n-z\}$. In many applications, there may be an overhead associated with excessive divisions into sub-tasks. We show that we can reduce the number of sub-tasks at the expense of a small increase of the master's waiting time. Using the so-called Δ-universal Staircase codes [25] reduces the number of sub-tasks at the expense of limiting the master to a set $\Delta \subseteq \{k, \dots, n\}$ of number of workers allowing the master to decode $A\mathbf{x}$ (see Section 2.6 for more details.) In other words, the master can decode $A\mathbf{x}$ by downloading enough information from any d workers, $d \in \Delta$. It remains to prove that d is concentrated around its mean. Hence, restricting d to a set Δ centered around its mean, leads to a reduction in the master's waiting time. Figure 3.3 depicts the concentration of d around its average for a $(100, 50, 1)$ system simulated on MATLAB. Figure 3.4, depicts the normalized difference between the mean waiting time of universal Staircase codes and Δ-universal Staircase codes for $(n, n/2, 1)$ systems with $\lambda = c = 1$ and $\Delta = \{d^* - 1, d^*, d^* + 1\}$, where d^* is the value of d that minimizes our upper bound in (3.5).

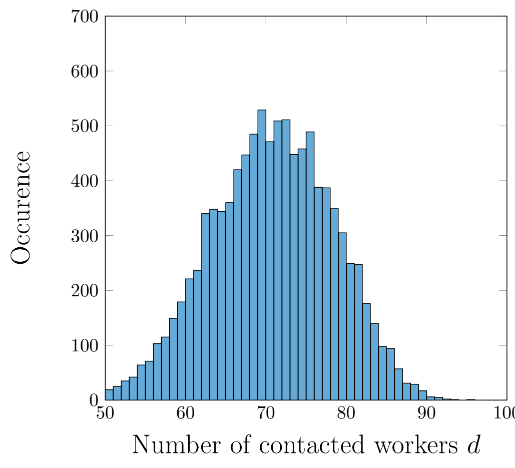


Figure 3.3: Histogram of the number of contacted workers for an $(n, k, z) = (100, 50, 1)$ system simulated on MATLAB over 10000 iterations with $\lambda = 1$ and $c = 1$.

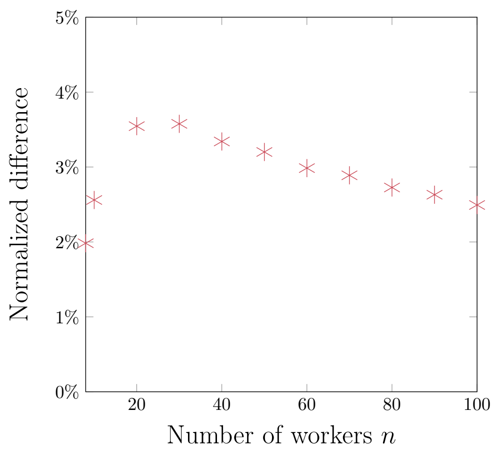


Figure 3.4: Normalized difference between the mean waiting time of universal Staircase codes and Δ-universal Staircase codes for systems with rate $k/n = 1/2$, $z = 1$, $\lambda = 1$ and $c = 1$ and $\Delta = \{d^* - 1, d^*, d^* + 1\}$, where d^* is the value of d that minimizes (3.5).

Next, we prove that the number of workers d that minimize the waiting time is concentrated around its average.

Lemma 3.6. *For an (n,k,z) system, the probability distribution of the distance between d and its average is*

$$\Pr\{|d-\mathbb{E}[d]|>t\}\le 2e^{-2t^2/n(n-k)^2}.$$

We prove Lemma 3.6 by showing that the number of workers d that first finish the aggregate computation is concentrated around its mean, using McDiarmid's inequality. Recall that $d:\mathbb{R}_+^n\to\{k,\dots,n\}$ is a function of the compute times $T_1,\dots,T_n$.

$$d(T_1,T_2,\dots,T_n)\triangleq \arg\min\left\{\frac{k-z}{i-z}T_{(i)}: i\in\{k,\dots,n\}\right\}.$$

Claim 3.7. *The number of workers d that minimize the waiting time is a bounded difference function of compute times with constants $(n-k,\dots,n-k)$. That is, for each $i\in[n]$ taking two vectors $\mathbf{t},\mathbf{t}^i\in\mathbb{R}_+^n$ that differ only in one position, i.e., $t_j=t_j^i$ for all $j\in[n]\setminus\{i\}$ and $t_i\neq t_i^i$,*

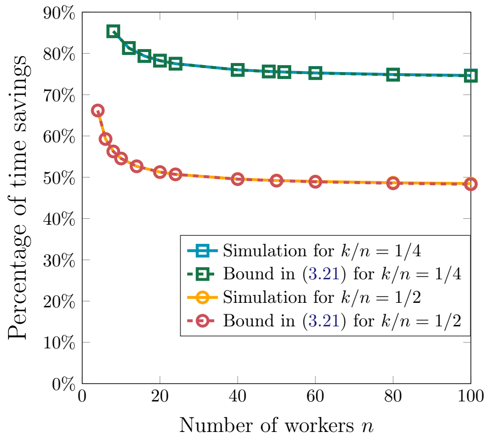
$$\sup\{|g(\mathbf{t})-g(\mathbf{t}^i)|:\mathbf{t},\mathbf{t}^i\in\mathbb{R}_+^n\}\le n-k. \tag{3.20}$$

The claim follows from the fact that $d\in\{k,\dots,n\}$. We prove the tightness of (3.20) using the following example.

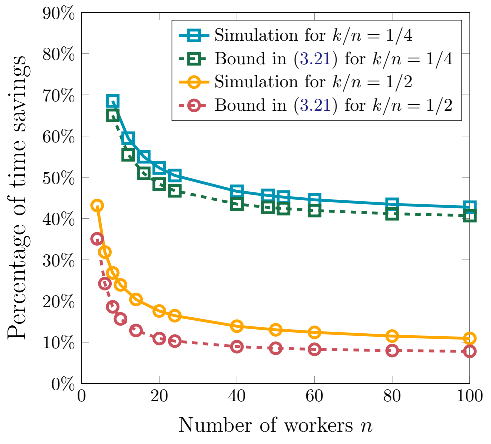
Example 3.3. *Consider the following realizations (ordered for simplicity) of $T_1,\dots,T_n$ of an (n,k,z) system, such that $T_k=t_k$, $T_i=t_i<t_k$ for $i=1,\dots,k-1$, and $t_i>(\frac{i-z}{k-z})t_k$ for $i=k+1,\dots,n$. The corresponding $g(t_1,\dots,t_n)$ is equal to k, because $t_k<(\frac{k-z}{i-z})t_i$ for all $k<i\le n$. Next, consider the ordered variables $(T_1,\dots,T_n')$ where only T_n changes to $t_n'\in(t_{n-1},(\frac{n-z}{k-z})t_k)$ while the other T_j's, $j\in\{1,\dots,n-1\}$, remain unchanged, then $g(t_1,\dots,t_n')=n$. We observe that the set $(t_{n-1},(\frac{n-z}{k-z})t_k)$ is not always empty since the condition $(\frac{k-z}{n-z-1})T_{n-1}>t_k$ only* *implies that* $(\frac{n-z-1}{k-z})t_k < T_{n-1} < T_n < (\frac{n-z}{k-z})t_k$. *Hence, there always exist a case where*

$$\sup_{\substack{t_1,\dots,t_n \in \mathbb{R}^n \\ t_i' \in \mathbb{R}}} |g(t_1,\dots,t_i,\dots,t_n) - g(t_1,\dots,t_i',\dots,t_n)| = n-k.$$

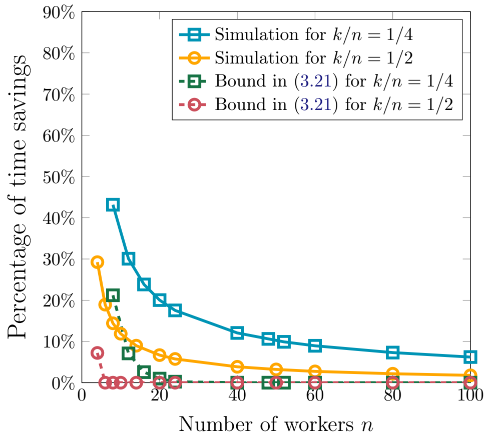
Therefore, we can apply the McDiarmid's inequality to obtain the concentration bound on d.



(a) Savings for $\lambda c = 100$.



(b) Savings for $\lambda c = 1$.



(c) Savings for $\lambda c = 0.001$.

Figure 3.5: Savings for the fixed rate regime, $k/n = 1/2$ and $1/4$. The lower bound on the savings of Staircase codes obtained from (3.21) is compared to the numerical values obtained by simulations. We consider systems with no colluding workers, i.e., $z = 1$, we fix $\lambda = 1$ and vary c. For instance, for systems with rate $k/n = 1/2$ and $\lambda c = 100$ Staircase codes can provide up to 66% reduction in the mean waiting time.

3.7 Simulations

We use the normalized difference between the mean waiting time of Staircase codes and classical secret sharing codes as a performance metric for Staircase codes. We refer to this metric as the savings. Using the result of Theorem 3.1, we can get a lower and an upper bound on the savings brought by Staircase codes. The lower bound on the savings is given in (3.21).

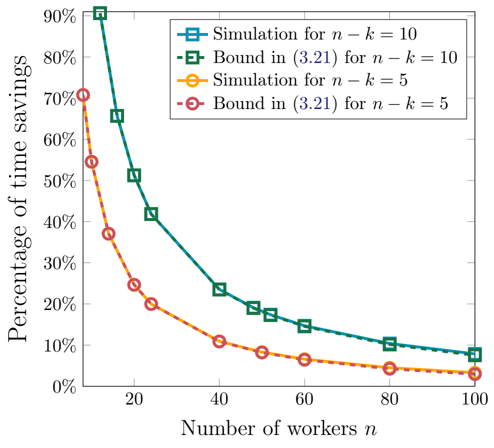
$$\frac{\mathbb{E}[T_{\mathrm{SS}}] - \mathbb{E}[T_{\mathrm{SC}}]}{\mathbb{E}[T_{\mathrm{SS}}]} \geq 1 - \min_{d \in \{k,\dots,n\}} \left\{ \frac{(k-z)(\lambda c + H_n - H_{n-d})}{(d-z)(\lambda c + H_n - H_{n-k})} \right\}. \tag{3.21}$$

To get an idea of the actual savings and the tightness of the bound in (3.21), we ran numerical simulations of the mean waiting time induced by the use of Staircase codes. By looking at (3.21), we notice that the bound depends on λ and c only through[2] λc (our simulations show that the actual savings also have a strong dependency on λc). Therefore, we consider three cases for λc : large values of λc ($\lambda c = 100$), medium values of λc ($\lambda c = 1$) and small values of λc ($\lambda c = 0.001$). We ran the simulations for two regimes:

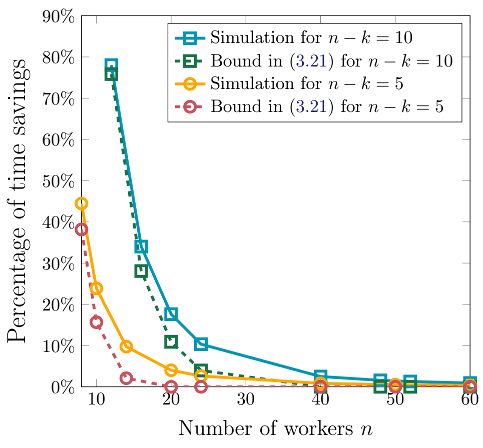
- *Fixed rate* k/n: the plots can be seen in Figure 3.5. We deduce from the plots that the lower bound is tighter for large values of λc. Moreover, the savings increase with the decrease of the rate k/n and the increase of λc. Note that for large values of λc, the lower bound in (3.21) converges to $1 - k/n$.

- *Fixed number of parities* $n - k$: the plots can be seen in Figure 3.6. We deduce from the plots that similarly to the fixed rate regime the lower bound is tight for large values of λc and that the savings increase with the increase of the number of parities $n - k$ and with the increase of λc. However, we observe that the savings vanish asymptotically with n in this regime.

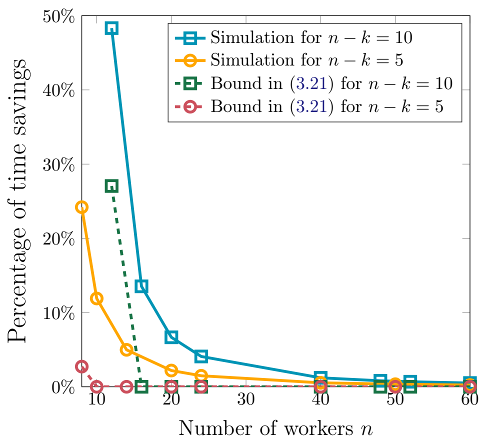
[2]Note that for $c = 0$ we go to the exponential model and the savings would depend only on λ.



(a) Savings for $\lambda c = 100$.



(b) Savings for $\lambda c = 1$.



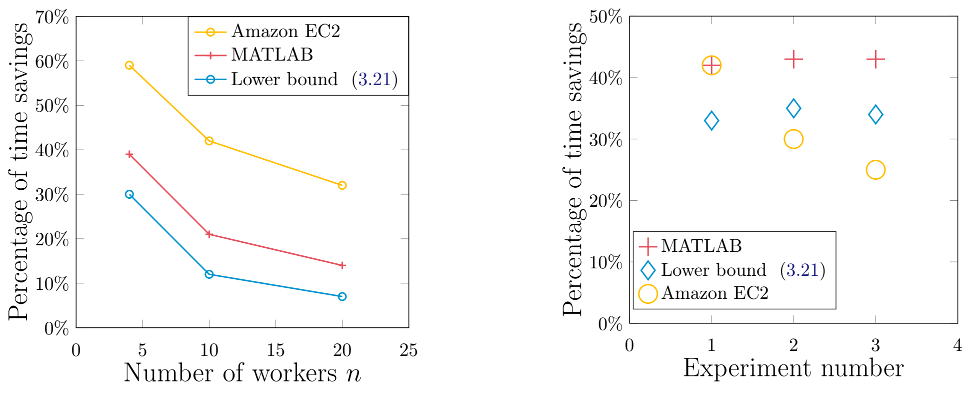
(c) Savings for $\lambda c = 0.001$.

Figure 3.6: Savings for the fixed number of parities regime, $n-k=5$ and 10. The lower bound on the savings of Staircase codes obtained from (3.21) is compared to the numerical values obtained by simulations. Similarly to Figure 3.5, we consider systems with $z=1$, $\lambda=1$ and vary c.

3.8 Implementation and Validation of the Theoretical Model

We describe a representative sample of our implementation on Amazon EC2 clusters and discuss our observations. In Section 3.8.1, we present traces for systems with fixed rate $k/n = 1/2$ (Figure 3.8). We noticed that the straggler behavior, and therefore the savings, can depend on the date and time of the implementation. To highlight this dependence, we present in Section 3.8.2 the traces of one system implemented at different date and times (Figure 3.9).

Discussion on the theoretical model: Before giving the details, we summarize our findings in Figure 3.7 that lists all the parameters that we implemented and compares the savings obtained



(a) Systems with $k = n/2$ and $z = 1$ implemented on Amazon EC2 instances. The data is a matrix of size 387000×250.

(b) $(4, 2, 1)$ system implemented three different times on Amazon EC2 instances. The data is a matrix of size 42000×250.

Figure 3.7: Comparison of the performance of Staircase codes on Amazon EC2 to the theoretical bound in (3.21) and the value obtained by simulations assuming the shifted exponential model in Section 3.2. To compute the bound, we measured the shift c^* and the rate λ^*, respectively, as the minimum response time and the inverse of the average response time at one worker over 1000 iterations.

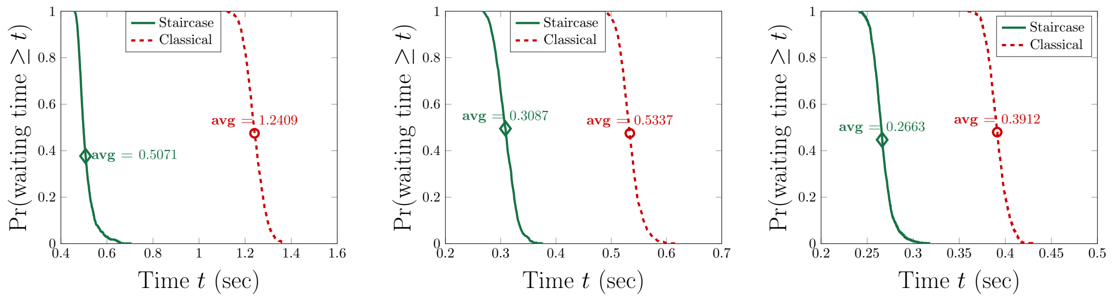
on Amazon to the theoretical lower bound (3.21) and numerical savings obtained by simulations. We observe that the savings of the system on EC2 can surpass the numerical values resulting from our theoretical model in Section 3.2 for large sizes of the matrix A. However, for small sizes of A, the savings in practice can be less.

The difference between the theoretical results and the implementations can be attributed to several reasons. First, in our model we assume in (4.6) that the total service time of a task does not change when divided into α sub-tasks, each requiring the same service time. Whereas, our implementation on Amazon shows that the download time decreases faster than linearly with the size of the sub-task for large sub-tasks. Second, for small sub-tasks, we noticed an additional overhead of sending the results of multiple sub-tasks. This overhead becomes non-negligible when the task is small. Third, we have assumed a homogeneous setting where all workers have the same behavior which is not always the case in practice.

Despite these differences, our adopted theoretical model is more amenable to theoretical analysis and provides insightful engineering guiding principles.

3.8.1 Implementation for systems with rate $k/n = 1/2$

We present the implementation of $(4, 2, 1)$, $(10, 5, 1)$ and $(20, 10, 1)$ systems on Amazon EC2 clusters. We use M4.large EC2 instances [78] from Amazon web services (AWS) for our implementation. We assign the master's job to an instance located in Virginia and the workers job to instances located in Ohio. We plot in Figures 3.8a, 3.8b and 3.8c the empirical complementary CDF of the master's waiting time for Staircase codes and classical secret sharing codes for $(4, 2, 1)$, $(10, 5, 1)$ and $(20, 10, 1)$ systems, respectively. The average savings brought by Staircase codes are 59%, 42% and 32% for systems with $n = 4$, $n = 10$ and $n = 20$ workers, respectively. These results are also summarized in Figure 3.7. Note that for this set of implementation, the master's data A is a matrix of size 378000×250 with entries generated uniformly at random from $\{1, \ldots, 255\}$. We run 1000 multiplications of A by a randomly generated vector $\mathbf{x}$.



(a) $(n, k, z) = (4, 2, 1)$.

(b) $(n, k, z) = (10, 5, 1)$.

(c) $(n, k, z) = (20, 10, 1)$.

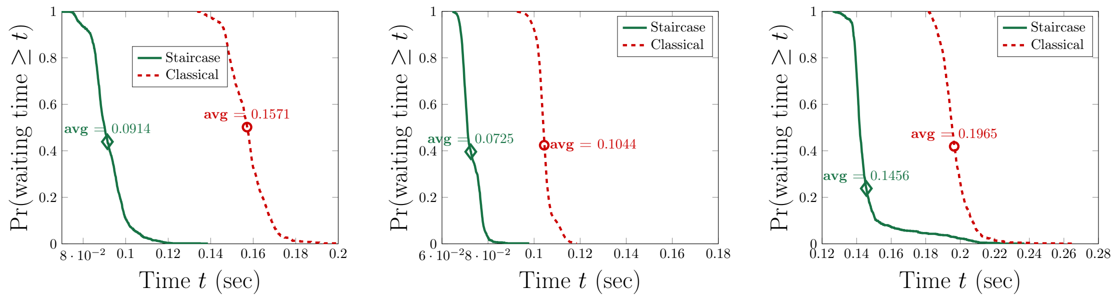
Figure 3.8: Empirical complementary CDF of the master's waiting time (and its average) observed on Amazon EC2 clusters for systems with rate $k/n = 1/2$. The data matrix A is a 378000×250 matrix with entries generated uniformly at random from $\{1, \ldots, 255\}$. Staircase codes bring 59% reduction in the mean waiting time for $n = 4$. Those numbers were obtained by repeating the multiplication process 1000 times.

3.8.2 Implementation on 4 worker instances at different times

We present the trace of a $(4, 2, 1)$ system implemented at different dates and times on Amazon EC2 clusters. We follow the same setting as before except that A is a 42000×250 matrix generated using the LFW dataset of public faces[3] [1]. We observe that the distribution of the

[3]To obtain the data matrix A, we convert the first 56 faces to 3 matrices each. Each matrix is a 250×250 matrix representing the color value of the pixels of each image in red, green and blue, respectively.

master's waiting time and the savings brought by using Staircase codes depend on the date and time of the implementation. This can be due to the varying state of the instances and the varying volume of traffic at Amazon servers.



(a) Tuesday 10-10-2017, 12:15 PM.

(b) Saturday 01-20-2018, 12:10 PM.

(c) Thursday 10-5-2017, 2:24 PM.

Figure 3.9: An $(n, k, z) = (4, 2, 1)$ system implemented on Amazon EC2 cluster at different times. The matrix A is a 42000×250 matrix representing 56 images from the LFW dataset [1]. We observe that the distribution of the master's waiting time and the savings brought by using Staircase codes (42%, 30%, and 25% respectively) depend on the date and time of the implementation.

3.9 Related Work

The problem of stragglers has been identified and studied by the distributed computing community, see e.g., [5, 39, 76, 79–93]. Recently, there has been a growing research interest in studying codes for straggler mitigation and delay minimization in distributed systems with no privacy constraints. The early body of work focused on minimizing latency of content download in distributed storage systems, see e.g., [26, 94–96] and later the focus has shifted to using codes for straggler mitigation in distributed computation.

In [6] Lee et al. studied the use of MDS codes for straggler mitigation in *linear* distributed machine learning algorithms. Yu et al. [97] introduced a coding scheme called polynomial codes to mitigate straggler in distributed matrix multiplication. Tandon et al. [10] introduced a framework called gradient coding for straggler mitigation in distributed gradient-descent based algorithms. In the same spirit of work, Halbawi et al. [11] proposed a gradient coding scheme that decreases the decoding complexity at the master.

In [7], Dutta et al. proposed new coding techniques that reduce the computation time at the workers side while accounting for stragglers. Moreover, coded computation was studied for specific applications, such as coded convolution [98] and coded linear transformations [99]. In a related context, Li et al. [100, 101] showed a fundamental tradeoff between the workers' computation load and the communication complexity in coded computation.

Secure multiparty computation [4] can be used in this setting to provide privacy. However, the methods there are generic and not tailored to to matrix multiplication and therefore do not have efficient communication cost and flexible straggler mitigation. The work that is closest to ours is [18] that studies the problem of distributively multiplying two private matrices under information theoretic privacy constraints using classical secret sharing codes. The problem of distributively multiplying two private matrices and the problem of running private distributed machine learning algorithm in the presence of stragglers has been further studied after the appearance of the original manuscript of this work [27] in [19–21,102] and [103–105] respectively.

In general, privacy in distributed computation is studied separately, mostly in the computer science community. Our work can also be related to the work on privacy-preserving algorithms, e.g., [106–109]. However, the privacy constraint in this line of work is computational privacy, and the proposed algorithms are not designed for straggler mitigation.

3.10 Conclusion and Open Problems

We consider the problem of private coded computation. We propose the use of a new family of secret sharing codes called Staircase codes that reduces the delays caused by stragglers. We show that Staircase codes always lead to smaller waiting time compared to classical secret sharing codes, e.g., Shamir secret sharing codes. The reason behind reducing the delays is that Staircase codes allow flexibility in the number of stragglers up to a given maximum, and universally achieve the information theoretic limit on the download cost by the master, leading to latency reduction. We consider the shifted exponential model for the workers's response time. In our analysis, we find upper and lower bounds on the master's mean waiting time.

We characterize the distribution of the master's waiting time, and its mean, for systems with $n = k - 1$ and $n = k - 2$. For general (n, k, z) systems. Moreover, we derive an expression that can give the exact distribution, and the mean, of the waiting time of the master. We supplement our theoretical study with extensive implementation on Amazon EC2 clusters.

While Staircase codes reduce the master's waiting time by minimizing the download cost, they are not designed to minimize latency. The problem of designing codes that minimize the latency remains open in general. Another open problem, which we leave for future work, is when malicious workers corrupt the results sent to the master.

Chapter 4

Adaptive Private Coded Computation

We consider the case where the workers are have different and time-varying resources, i.e., the workers have different computation and communication specifications that can change with time. Resources of the workers depend on many factors such as the geographic location of the workers, network congestion, computation workload and worker battery life. Applications include clusters with variability in the computation workload and network congestion at the workers. Other applications include Internet of Things networks and Edge computing where the workers also enjoy great variability due to the difference in the nature of the devices (phone, tablet, sensor, etc).

Our goal is to design coding schemes that adapt to the variability of the resources at the workers and reduce the waiting time at the master. Our key tool is the theory of coded computation, which advocates mixing data in computationally intensive tasks by employing erasure codes and offloading these tasks to the workers. We develop a private and rateless adaptive coded computation (PRAC) algorithm for private coded matrix-vector multiplication by taking into account the privacy requirements and the heterogeneous and time-varying resources of the workers. PRAC is based on Fountain codes coupled with an MDS code to ensure privacy. We show that in this setting PRAC outperforms known secure coded computation methods. We provide theoretical guarantees on the performance of PRAC and its comparison to baselines. Moreover, we supplement our theoretical results with extensive simulations and implementation on Android devices.

4.1 Introduction

Edge computing is emerging as a new paradigm to allow processing data near the edge of the network, where the data is typically generated and collected. This enables computation at the edge in applications such as Internet of Things (IoT), in which an increasing number of devices (sensors, cameras, health monitoring devices, etc.) collect data that needs to be processed through computationally intensive algorithms with stringent reliability, security and latency constraints.

One of the promising solutions to handle computationally intensive tasks is computation offloading, which advocates offloading tasks to remote servers or cloud. Yet, offloading tasks to remote servers or cloud could be luxury that cannot be afforded by most of the edge applications, where connectivity to remote servers can be lost or compromised, which makes edge computing crucial.

Edge computing advocates that computationally intensive tasks in a device (master) could be offloaded to other edge or end devices (workers) in close proximity. However, offloading tasks to other devices leaves the IoT and the applications it is supporting at the complete mercy of an attacker. Furthermore, exploiting the potential of edge computing is challenging mainly due to the heterogeneous and time-varying nature of the devices at the edge. Thus, our goal is to develop a private, dynamic, adaptive, and heterogeneity-aware cooperative computation framework that provides both privacy and computation efficiency guarantees. Note that the application of this work can be extended to cloud computing at remote data-centers. However, we focus on edge computing as heterogeneity and time-varying resources are more prevalent at the edge as compared to data-centers.

Our key tool is the theory of coded computation, which advocates mixing data in computationally intensive tasks by employing erasure codes and offloading these tasks to other devices for computation [6,7,10,11,27,97–101,110–112]. The following canonical example demonstrates the effectiveness of coded computation.

Example 4.1. *Consider the setup where a master device wishes to offload a task to* 3 *workers. The master has a large data matrix* A *and wants to compute a matrix vector product* $A\mathbf{x}$*. The master device divides the matrix* A *row-wise equally into two smaller matrices* A_1 *and* A_2*, which are then encoded using a* $(3, 2)$ *Maximum Distance Separable (MDS) code to give* $S_1 = A_1$*,* $S_2 = A_2$ *and* $S_3 = A_1 + A_2$*. The master sends each "share"* S_i *to worker* i*,* $i \in \{1, 2, 3\}$*. Also, the master sends* $\mathbf{x}$ *to the workers and ask them to compute* $S_i\mathbf{x}$*,* $i \in \{1, 2, 3\}$*. When the master receives the computed values (i.e.,* $S_i\mathbf{x}$*) from at least two out of three workers, it can decode its desired task, which is the computation of* $A\mathbf{x}$*. The power of coded computations is that it makes* $S_3 = A_1 + A_2$ *act as a "joker" redundant task that can replace any of the other two tasks if they end up straggling or failing.*

The above example demonstrates the benefit of coding for edge computing. However, the very nature of task offloading from a master to worker devices makes the computation framework vulnerable to attacks. One of the attacks, which is also the focus of this Chapter, is *eavesdropper adversary*, where one or more workers can behave as an eavesdropper and can spy on the coded data sent to these devices for computations.[1] For example, $S_3 = A_1 + A_2$ in Example 4.1 can be processed and spied by worker 3. Even though $A_1 + A_2$ is coded, the attacker can infer some information from this coded task. Privacy against eavesdropper attacks is extremely important in edge computing [113–115]. Thus, it is crucial to develop a private coded computation mechanism against eavesdropper adversary who can gain access to offloaded tasks.

[1]Note that there are other types of attacks; for example *Byzantine adversary*, which are out of scope of this work.

Time	Worker 1	Worker 2	Worker 3
1	R_1	$\mathbf{A_1} + \mathbf{A_3} + R_1$	$\mathbf{A_3} + R_1$
2	▨	R_2	▨
3	$\mathbf{A_2} + \mathbf{A_3} + R_2$	▨	▨
4	▨	▨	$\mathbf{A_2} + R_2$

Table 4.1: Example of PRAC operation in heterogeneous and time-varying setup for $n = 3$ workers.

We develop a private and rateless adaptive coded computation (PRAC) mechanism. PRAC is (i) private as it is secure against eavesdropper adversary, (ii) rateless, because it uses Fountain codes [29–31] instead of Maximum Distance Separable (MDS) codes [116,117], and (iii) adaptive as the master device offloads tasks to workers by taking into account their heterogeneous and time-varying resources. Next, we illustrate the main idea of PRAC through an illustrative example.

Example 4.2. *We consider the same setup as in Example 4.1, where a master device offloads tasks to* 3 *workers. The master has a large data matrix* A *and wants to compute matrix vector product* $A\mathbf{x}$*. The master divides the matrix* A *row-wise into* 3 *sub-matrices* A_1*,* A_2*,* A_3*; and encodes these matrices using a Fountain code*[2] *[29–31]. An example set of coded packets is* A_2*,* A_3*,* $A_1 + A_3$*, and* $A_2 + A_3$*. However, prior to sending a coded packet to a worker, the master generates a random key matrix* R *with the same dimensions as* A_i *and with entries drawn uniformly at random from the same alphabet as the entries of* A*. The key matrix is added to the coded packets to provide privacy as shown in Table 4.1. In particular, a key matrix* R_1 *is created at the start of time slot* 1*, combined with* $A_1 + A_3$ *and* A_3*, and transmitted to workers* 2 *and* 3*, respectively.* R_1 *is also transmitted to worker* 1 *in order to obtain* $R_1\mathbf{x}$ *that will help the master in the decoding process. The computation of* $(A_1 + A_3 + R_1)\mathbf{x}$ *is completed at the end of time slot* 1*. Thus, at that time slot the master generates a new matrix,* R_2*, and sends it to worker* 2*. At the end of time slot* 2*, worker* 1 *finishes its computation, therefore the master adds* R_2 *to* $A_2 + A_3$ *and sends it to worker* 1*. A similar process is repeated at the end of time slot* 3*. Now the master waits for worker* 2 *to return* $R_2\mathbf{x}$ *and for any other worker to*

[2]Fountain codes are desirable here for two properties: (i) they provide a fluid abstraction of the coded packets so the master can always decode with high probability as long as it collects enough packets; (ii) They have low decoding complexity.

return its uncompleted task in order to decode $A\mathbf{x}$. Thanks to using key matrices R_1 and R_2, and assuming that workers do not collude, privacy is guaranteed. On a high level, privacy is guaranteed because the observation of the workers is statistically independent from the data A.

This example shows that PRAC can take advantage of coding for computation, and provide privacy.

Contributions. We design PRAC for heterogeneous and time-varying private coded computation with colluding workers. In particular, PRAC codes sub-tasks using Fountain codes, and determines how many coded packets and keys each worker should compute dynamically over time. We provide theoretical analysis of PRAC and show that it (i) guarantees privacy conditions, (ii) uses minimum number of keys to satisfy privacy requirements, and (iii) maintains the desired rateless property of non-private Fountain codes. Furthermore, we provide a closed form task completion delay analysis of PRAC. Finally, we evaluate the performance of PRAC via simulations and implementation on Android devices.

The use of Fountain codes in encoding the sub-tasks provides PRAC flexibility in the number of stragglers and on the computing capacity of workers, reflected by the number of sub-tasks assigned to each worker. In contrast, existing solutions for private coded computation require the master to set a threshold on the number of stragglers that it can tolerate and pre-assign the sub-tasks to the workers based on this threshold.

Organization. The structure of the rest of this Chapter is as follows. We start with presenting the system model in Section 4.2. Section 4.3 presents the design of private and rateless adaptive coded computation (PRAC). We characterize and analyze PRAC in Section 4.4. We present evaluation results in section 4.5. Section 5.8 presents related work. Section 4.7 concludes the Chapter.

4.2 System Model

Setup. We consider a master/workers setup at the edge of the network, where the master device

$\mathtt{M}$ offloads its computationally intensive tasks to workers $\mathtt{W}_i,\ i \in \mathcal{N}$, (where $|\mathcal{N}| = n$) via device-to-device (D2D) links such as Wi-Fi Direct and/or Bluetooth. The master device divides a task into smaller sub-tasks, and offloads them to workers that process these sub-tasks in parallel.

Task Model. We focus on the computation of linear functions, i.e., matrix-vector multiplication. We suppose the master wants to compute the matrix vector product $A\mathbf{x}$, where $A \in \mathbb{F}_q^{m\times\ell}$ can be thought of as the data matrix and $\mathbf{x} \in \mathbb{F}_q^{\ell}$ can be thought of as an attribute vector. We assume that the entries of A and $\mathbf{x}$ are drawn independently and uniformly at random[3] from $\mathbb{F}_q$. The motivation stems from machine learning applications where computing linear functions is a building block of several iterative algorithms [118, 119]. For instance, the main computation of a gradient descent algorithm with squared error loss function is

$$\mathbf{x}^+ = \mathbf{x} - \gamma A^T(A\mathbf{x} - \mathbf{y}), \tag{4.1}$$

where $\mathbf{x}$ is the value of the attribute vector at a given iteration, $\mathbf{x}^+$ is the updated value of $\mathbf{x}$ at this iteration and the learning rate γ is a parameter of the algorithm. Equation (4.1) consists of computing two linear functions $A\mathbf{x}$ and $A^T\mathbf{w} \triangleq A^T(A\mathbf{x} - \mathbf{y})$.

Worker and Attack Model. The workers incur random delays while executing the task assigned to them by the master device. The workers have different computation and communication specifications resulting in a heterogeneous environment which includes workers that are significantly slower than others, known as stragglers. Moreover, the workers cannot be trusted with the master's data. We consider *eavesdropper adversaries*, where one or more of workers can be eavesdroppers and can spy on the coded data sent to these devices for computations. We assume that up to z, $z < n$, workers can collude, i.e., z workers can share the data they received from the master in order to obtain information about A. The parameter z can be chosen based on the desired privacy level; a larger z means a higher privacy level and vice versa. One would want to set z to the largest possible value for maximum, $z = n - 1$ security purposes. However,

[3]We abuse notation and denote both the random matrix representing the data and its realization by A. We do the same for $\mathbf{x}$.

this has the drawback of increasing the complexity and the runtime of the algorithm. In our setup we assume that z is a fixed and given system parameter.

Coding & Secret Keys. The matrix A can be divided into b row blocks (we assume that b divides m, otherwise all-zero rows can be added to the matrix to satisfy this property) denoted by A_i, $i = 1, \dots, b$. The master applies Fountain coding [29–31] across row blocks to create information packets $U_j \triangleq \sum_{i=1}^{m} c_{i,j} A_i$, $j = 1, 2, \dots$, where $c_{i,j} \in \{0, 1\}$. Note that an information packet is a matrix of dimension $m/b \times \ell$, i.e., $U_j \in \mathbb{F}_q^{m/b \times \ell}$. Such rateless coding is compatible with our goal to create adaptive coded cooperation computation framework.

In order to maintain privacy of the data, the master device generates random matrices R_i of dimension $m/b \times \ell$ called *keys.* The entries of the R_i matrices are drawn uniformly at random from the same field as the entries of A. Each information packet U_j is *padded* with a linear combination of z keys $f_j(R_{i,1}, \dots, R_{i,z})$ to create a secure packet $S_j \in \mathbb{F}_q^{m/b \times \ell}$ defined as $S_j \triangleq U_j + f_j(R_{i,1}, \dots, R_{i,z})$.

The master sends $\mathbf{x}$ to all the workers, then it sends the keys and the S_j's to the workers according to our PRAC scheme described later. Each worker multiplies the received packet by $\mathbf{x}$ and sends the result back to the master. Since the encoding is rateless, the master keeps sending packets to the workers until it can decode $A\mathbf{x}$. The master then sends a stop message to all the workers.

Privacy Conditions. Our primary requirement is that any collection of z (or less) workers will not be able to obtain any information about A, in an information theoretic sense.

In particular, let P_i, $i = 1 \dots, n$, denote the collection of packets sent to worker $\mathtt{W}_i$. For any set $\mathcal{B} \subseteq \{1, \dots, n\}$, let $P_{\mathcal{B}} \triangleq \{P_i, i \in \mathcal{B}\}$ denote the collection of packets given to worker $\mathtt{W}_i$ for all $i \in \mathcal{B}$. Let $\mathtt{A}$ denote the random variable representing the matrix A and $\mathtt{P}_{\mathcal{B}}$ denotes the collection of random variables representing the packets given to worker $\mathtt{W}_i$ for all $i \in \mathcal{B}$. The

Symbol	Meaning	Symbol	Meaning
$\mathtt{M}$	master	R	random matrix
$\mathtt{W}_i$	worker i	U	Fountain coded packet of A_i's
n	number of workers	S	secure Fountain coded packet
z	number of colluding workers	T_i	time to compute a packet at $\mathtt{W}_i$
A	$m \times \ell$ data matrix	$T_{(d)}$	d^{th} order statistic of T_i's
A_i	i^{th} row block of data matrix A	T	time spent by $\mathtt{M}$ to decode $A\mathbf{x}$
m	number of rows in A	RTT_i	time to communicate a packet at $\mathtt{W}_i$
b	number of row blocks in A	$\tau_{t,i}$	computation time of the t^{th} packet at $\mathtt{W}_i$
$\mathbf{x}$	$\ell \times 1$ attribute vector	ε	overhead of Fountain codes
P_i	collection of all packets sent to $\mathtt{W}_i$	$P_{t,i}$	t^{th} packet sent to $\mathtt{W}_i$

Table 4.2: Summary of notations for this Chapter.

privacy requirement[4] can be expressed as

$$H(\mathtt{A}|\mathtt{P}_{\mathcal{Z}}) = H(\mathtt{A}), \quad \forall \mathcal{Z} \subseteq \{1, \dots, n\} \text{ s.t. } |\mathcal{Z}| \leq z. \tag{4.2}$$

$H(\mathtt{A})$ denotes the entropy, or uncertainty, about the random variable $\mathtt{A}$ and $H(\mathtt{A}|\mathtt{P}_{\mathcal{Z}})$ denotes the uncertainty about the random variable $\mathtt{A}$ after observing the collection of packets $P_{\mathcal{Z}}$.

Delay Model. Each packet transmitted from the master to a worker $\mathtt{W}_i$, $i = 1, 2, ..., n$, experiences the following delays: (i) transmission delay for sending the packet from the master to the worker, (ii) computation delay for computing the multiplication of the packet by the vector $\mathbf{x}$, and (iii) transmission delay for sending the computed packet from worker $\mathtt{W}_i$ back to the master. We denote by $\tau_{t,i}$ the computation time of the t^{th} packet at worker $\mathtt{W}_i$ and RTT_i denotes the average round-trip time spent to send and receive a packet from worker $\mathtt{W}_i$. The time spent by the master is equal to the time taken by the $(z+1)^{\text{st}}$ fastest worker to finish its assigned tasks.

[4]In some cases the vector $\mathbf{x}$ may contain information about A and therefore must not be revealed to the workers. We explain in Appendix A how to generalize our scheme to account for such cases.

4.3 Design of PRAC

4.3.1 Overview

We present the detailed explanation of PRAC. Let $P_{t,i} \in \mathbb{F}_q^{m/b\times\ell}$ be the t^{th} packet sent to worker $\mathtt{W}_i$. This packet can be either a key or a secure packet. For each value of t, the master sends z keys denoted by $R_{t,1}, \dots, R_{t,z}$ to z different workers and up to $n-z$ secure packets $S_{t,1}, \dots, S_{t,n-z}$ to the remaining workers. The master needs the results of $b+\varepsilon$ information packets, i.e., $U_{t,i}\mathbf{x}$, to decode the final result $A\mathbf{x}$, where ε is the overhead required by Fountain coding[5]. To obtain the results of $b+\varepsilon$ information packets, the master needs the results of $b+\varepsilon$ secure packets, $S_{t,i}\mathbf{x} = (U_{i,j} + f_j(R_{t,i}, \dots, R_{t,z}))\mathbf{x}$, together with all the corresponding[6] $R_{t,i}\mathbf{x}, i = 1, \dots, z$. Therefore, only the results of the $S_{t,i}\mathbf{x}$ for which all the computed keys $R_{t,i}\mathbf{x}, i = 1, ..., z$, are received by the master can account for the total of $b+\varepsilon$ information packets.

4.3.2 Dynamic rate adaptation

The dynamic rate adaptation part of PRAC is based on [110]. In particular, the master offloads coded packets gradually to workers and receives two acknowledgements (ACKs) for each transmitted packet; one confirming the receipt of the packet by the worker, and the second one (piggybacked to the computed packet) showing that the packet is computed by the worker. Then, based on the frequency of the received ACKs, the master decides to transmit more/less coded packets to that worker. In particular, each packet $P_{t,i}$ is transmitted to each worker $\mathtt{W}_i$ before or right after the computed packet $P_{t-1,i}\mathbf{x}$ is received at the master. For this purpose, the average per packet computing time $\mathbb{E}[\tau_{t,i}]$ is calculated for each worker $\mathtt{W}_i$ dynamically based on the previously received ACKs. Each packet $P_{t,i}$ is transmitted after waiting $\mathbb{E}[\tau_{t,i}]$ from the time $P_{t-1,i}$ is sent or right after packet $P_{t-1,i}\mathbf{x}$ is received at the master, thus reducing the idle

[5]The overhead required by Fountain coding is typically as low as 5% [31], i.e., $\varepsilon = 0.05b$

[6]Recall that $f_j(R_{t,1}, \dots, R_{t,z})$ is a linear function, thus it is easy to extract $(R_{t,i})\mathbf{x}, i = 1, ..., z$, from $(f_j(R_{t,1}, \dots, R_{t,z}))\mathbf{x}$.

Time	Worker 1	Worker 2	Worker 3	Worker 4
1	$R_{1,1}$	$R_{1,2}$	$\mathbf{A_4} + R_{1,1} + R_{1,2}$	$\mathbf{A_3} + \mathbf{A_4} + \mathbf{A_6} + R_{1,1} + 2R_{1,2}$
2				$R_{2,1}$
3	$R_{2,2}$			
4		$\mathbf{A_3} + R_{2,1} + R_{2,2}$	$\mathbf{A_4} + \mathbf{A_5} + R_{2,1} + 2R_{2,2}$	
5		$R_{3,1}$		
6	$\mathbf{A_2} + R_{3,1} + R_{3,2}$			$R_{3,2}$
7		$R_{4,1}$	$\mathbf{A_1} + R_{3,1} + 2R_{3,2}$	
8	$R_{4,2}$			$\mathbf{A_2} + \mathbf{A_3} + R_{4,1} + R_{4,2}$

Table 4.3: Depiction of PRAC in the presence of stragglers. The master keeps generating packets using Fountain codes until it can decode $A\mathbf{x}$. The master estimates the average task completion time of each worker and sends a new packet to avoid idle time. Each new packet sent to a worker must be secured with a new random key. The master can decode $A_1\mathbf{x}, \ldots, A_6\mathbf{x}$ after receiving all the packets not having $R_{4,1}$ or $R_{4,2}$ in them.

time at the workers. This policy is shown to approach the optimal task completion delay and maximizes the workers' efficiency and is shown to improve task completion time significantly compared with the literature [110].

4.3.3 Coding

We explain the coding scheme used in PRAC. We start with an example to build an intuition and illustrate the scheme before going into details.

Example 4.3. *Assume there are $n = 4$ workers out of which any $z = 2$ can collude. Let A and $\mathbf{x}$ be the data owned by the master and the vector to be multiplied by A, respectively. The master sends $\mathbf{x}$ to all the workers. For the sake of simplicity, assume A can be divided into $b = 6$ row blocks, i.e., $A = \begin{bmatrix} A_1^T & A_2^T & \ldots & A_6^T \end{bmatrix}^T$. The master encodes the A_i's using Fountain code. We denote by* round *the event when the master sends a new packet to a worker. For example, we say that worker 1 is at round 3 if it has received 3 packets so far. For every round t, the master generates $z = 2$ random matrices $R_{t,1}$, $R_{t,2}$ (with the same size as A_1) and encodes them using an $(n, z) = (4, 2)$ systematic maximum distance separable (MDS) code by multiplying $R_{t,1}$, $R_{t,2}$ by a generator matrix G as follows*

$$G \begin{bmatrix} R_{t,1} \\ R_{t,2} \end{bmatrix} \triangleq \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} R_{t,1} \\ R_{t,2} \end{bmatrix}. \tag{4.3}$$

This results in the encoded matrices of $R_{t,1}$, $R_{t,2}$, $R_{t,1} + R_{t,2}$, and $R_{t,1} + 2R_{t,2}$. Now let us assume that workers can be stragglers. At the beginning, the master initializes all the workers at round 1. Afterwards, when a worker $\mathtt{W}_i$ finishes its task, the master checks how many packets this worker has received so far and how many other workers are at least at this round. If this worker $\mathtt{W}_i$ is the first or second to reach round t, the master generates $R_{t,1}$ or $R_{t,2}$, respectively, and sends it to $\mathtt{W}_i$. Otherwise, if $\mathtt{W}_i$ is the j^{th} worker ($j > 2$) to reach round t, the master multiplies $\begin{bmatrix} R_{t,1} & R_{t,2} \end{bmatrix}^T$ by the j^{th} row of G, adds it to a generated Fountain coded packet, and sends it to $\mathtt{W}_i$. The master keeps sending packets to the workers until it can decode $A\mathbf{x}$. We illustrate the idea in Table 4.3.

We now explain the details of PRAC in the presence of z colluding workers.

1. *Initialization:* The master divides A into b row blocks $A_1, \dots, A_b$ and sends the vector $\mathbf{x}$ to the workers. Let $G \in \mathbb{F}_q^{n \times z}$, $q > n$, be the generator matrix of an (n, z) systematic MDS code. For example one may use systematic Reed-Solomon codes that use Vandermonde matrix as generator matrix, see for example [120]. The master generates z random matrices $R_{1,1}, \dots, R_{1,z}$ and encodes them using G. Each coded key can be denoted by $\mathbf{g}_i \mathcal{R}_1$ where $\mathbf{g}_i$ is the i^{th} row of G and $\mathcal{R}_t \triangleq \begin{bmatrix} R_{t,1}^T & \dots & R_{t,z}^T \end{bmatrix}^T$. The master sends the z keys $R_{1,1}, \dots, R_{1,z}$ to the first z workers, generates $n - z$ Fountain coded packets of the A_i's, adds to each packet an encoded random key $\mathbf{g}_i \mathcal{R}_1$, $i = z + 1, \dots n$, and sends them to the remaining $n - z$ workers.

2. *Encoding and adaptivity:* When the master wants to send a new packet to a worker (noting that a packet $P_{t,i}$ is transmitted to worker $\mathtt{W}_i$ before, or right after, the computed packet $P_{t-1,i}\mathbf{x}$ is received at the master according to the strategy described in Section 4.3.2), it checks at which round this worker is, i.e., how many packets this worker has received so far, and checks how many other workers are at least at this round. Assume worker $\mathtt{W}_i$ is at round t and $j - 1$ other workers are at least at this round. If $j \leq z$, the master generates and sends $R_{t,j}$ to the worker. However, if $j > z$ the master generates a Fountain coded packet of the A_i's (e.g., $A_1 + A_2$), adds to it $\mathbf{g}_j \mathcal{R}_t$ and sends the packet $(A_1 + A_2 + \mathbf{g}_j \mathcal{R}_t)$

to the worker. Each worker computes the multiplication of the received packet by the vector $\mathbf{x}$ and sends the result to the master.

3. *Decoding and speed:* Let p_i denote the number of packets sent to worker $\mathtt{W}_i$. We define $p_{max} \triangleq \max_i p_i$ such that at the end of the process the master has $R_{t,i}\mathbf{x}$ for all $t = 1, \dots, p_{max}$ and all $i = 1, \dots, z$. The master can therefore subtract $R_{t,i}$, $t = 1, \dots, p_{max}$ and $i = 1, \dots, z$, from all received secure information packets, and thus can decode the A_i's using the Fountain code decoding process. The number of secure packets that can be used to decode the A_i's is dictated by the $(z+1)^{\text{st}}$ fastest worker, i.e., the master can only use the results of secure information packets computed at a given round if at least $z+1$ workers have completed that round. If for example the z fastest workers have completed round 100 and the $(z+1)^{\text{st}}$ fastest worker has completed round 20, the master can only use the packets belonging to the first 20 rounds. The reason is that the master needs all the keys corresponding to a given round in order to use the secure information packet for decoding. In Lemma 4.4 we prove that this scheme is optimal, i.e., in private coded computation the master cannot use the packets computed at rounds finished by less than $z+1$ workers irrespective of the coding scheme.

4.4 Performance Analysis of PRAC

4.4.1 Privacy

In this section, we provide theoretical analysis of PRAC by particularly focusing on its privacy properties.

Theorem 4.1. *PRAC is a rateless real-time adaptive coded computation scheme that allows a master device to run distributed linear computation on private data A via n workers while satisfying the privacy constraint given in (4.2) for a given $z < n$.*

Proof. Since the random keys are generated independently at each round, it is sufficient to study the privacy of the data on one round and the privacy generalizes to the whole algorithm. Let t be a given round of PRAC. We show that for any subset $\mathcal{Z} \subset \{1, \dots, n\}, |\mathcal{Z}| = z$, the collection of packets $P_{\mathcal{Z}} \triangleq \{P_{t,i}, i \in \mathcal{Z}\}$ sent at round t reveals no information about the data A as given in (4.2), i.e., $H(\mathtt{A}) = H(\mathtt{A}|\mathtt{P}_{\mathcal{Z}})$. Let $\mathtt{R}$ denote the random variable representing all the keys generated at round t, then it is enough to show that $H(\mathtt{R}|\mathtt{A}, \mathtt{P}_{\mathcal{Z}}) = 0$ as detailed in Appendix C.1 (a similar proof is given in Lemma 2.4). Therefore, we need to show that given A as side information, any z workers can decode the random keys $R_{t,1}, \dots, R_{t,z}$. Without loss of generality assume the workers are ordered from fastest to slowest, i.e., worker $\mathtt{W}_1$ is the fastest at the considered round t. Since the master sends z random keys to the fastest z workers, then $P_{t,i} = R_{t,i}, i = 1, \dots, z$. The remaining $n - z$ packets are secure information packets sent to the remaining $n - z$ workers, i.e., $P_{t,i} = S_{t,i} = U_{t,i} + f(R_{t,1}, \dots, R_{t,z})$, where $U_{t,i}$ is a linear combination of row blocks of A and $f(R_{t,1}, \dots, R_{t,z})$ is a linear combination of the random keys generated at round t. Given the data A as side information, any collection of z packets can be expressed as z codewords of the (n, z) MDS code encoding the random keys. Thus, given the matrix A, any collection of z packets is enough to decode all the keys and $H(\mathtt{R}|\mathtt{A}, \mathtt{P}_{\mathcal{Z}}) = 0$ which concludes the proof. □

Remark 4.2. PRAC requires the master to wait for the $(z+1)^{\text{st}}$ fastest worker in order to be able to decode $A\mathbf{x}$. We show in Lemma 4.4 that this limitation is a byproduct of all private coded computation schemes.

Remark 4.3. PRAC uses the minimum number of keys required to guarantee the privacy constraints. At each round PRAC uses exactly z random keys which is the minimum amount of required keys (c.f. Equation (C.5) in Appendix C.1).

Lemma 4.4. *Any private coded computation scheme for distributed linear computation limits the master to the speed of the $(z+1)^{st}$ fastest worker.*

Proof. We prove the lemma by contradiction. Assume that there exists a private coded computation scheme for distributed linear computation that is secure against z colluding workers and allows the master to decode $A\mathbf{x}$ using the help of the fastest z workers. Without loss of generality, assume that the workers are ordered from the fastest to the slowest, i.e., worker $\mathtt{W}_1$ is the fastest and worker $\mathtt{W}_n$ is the slowest. The previous assumption implies that the results sent from the first z workers contain information about $A\mathbf{x}$, otherwise the master would have to wait for at least the $(z+1)^{\text{st}}$ fastest worker to decode $A\mathbf{x}$. By linearity of the multiplication $A\mathbf{x}$, decoding information about $A\mathbf{x}$ from the results of z workers implies decoding information about A from the packets sent to those z workers. Hence, there exists a set $\mathcal{Z} \subset \{1, \dots, n\}$ of z workers for which $H(\mathtt{A}|\mathtt{P}_{\mathcal{Z}}) \neq 0$, where $\mathtt{P}_{\mathcal{Z}}$ denotes the random variable representing the tasks allocated to those z workers, hence violating the privacy constraint. Therefore, any private coded computation scheme for linear computation limits the master to the speed of the $(z+1)^{\text{st}}$ fastest worker in order to decode the wanted result. □

4.4.2 Task completion delay

In this section, we characterize the task completion delay of PRAC and compare it with Staircase codes [27] (explained in Chapter 2 and Chapter 3), which are secure against eavesdropping attacks in a coded computation setup with homogeneous resources. First, we start with task completion delay characterization of PRAC.

Theorem 4.5. *Let b be the number of row blocks in A, let $\tau_{t,i}$ denote the computation time of the t^{th} packet at worker $\mathtt{W}_i$ and let RTT_i denote the average round-trip time spent to send and receive a packet from worker $\mathtt{W}_i$. The task completion time of PRAC is approximated as*

$$T_{PRAC} \approx \max_{i \in \{1,\dots,n\}} \{RTT_i\} + \frac{b+\varepsilon}{\sum_{i=z+1}^{n} 1/\mathbb{E}[\tau_{t,i}]}, \tag{4.4}$$

$$\approx \frac{b+\varepsilon}{\sum_{i=z+1}^{n} 1/\mathbb{E}[\tau_{t,i}]}, \tag{4.5}$$

where $\mathtt{W}_i$'s are ordered indices of the workers from fastest to slowest, i.e., $\mathtt{W}_1 = \arg\min_i \mathbb{E}[\tau_{t,i}]$.

Proof. The proof of Theorem 4.5 is given in Appendix C.2 . □

Now that we characterized the task completion delay of PRAC, we can compare it with the state-of-the-art. Private coded computation schemes that exist in the literature usually use static task allocation, where tasks are assigned to workers a priori.

The most recent work in the area is Staircase codes, which is shown to outperform all existing schemes that use threshold secret sharing [27]. However, Staircase codes are static; they allocate fixed amount of tasks to workers a priori. Thus, Staircase codes cannot leverage the heterogeneity of the system, neither can it adapt to a system that is changing in time. On the other hand, our solution PRAC adaptively offloads tasks to workers by taking into account the heterogeneity and time-varying nature of resources at workers. Therefore, we restrict our focus on comparing PRAC to Staircase codes.

Staircase codes assigns a task of size $b/(k-z)$ row blocks to each worker.[7] Let T_i be the time spent at worker $\mathtt{W}_i$ to compute the whole assigned task. Denote by $T_{(i)}$ the i^{th} order statistic of the T_i's and by $T_{\text{SC}}(n,k,z)$ the task completion time, i.e., time the master waits until it can decode $A\mathbf{x}$, when using Staircase codes. In order to decode $A\mathbf{x}$ the master needs to receive a fraction equal to $(k-z)/(d-z)$ of the task assigned to each worker from any d workers where $k \leq d \leq n$. From (4.6) the task completion time of the master can be expressed as

$$T_{\text{SC}}(n,k,z) = \min_{d \in \{k,\dots,n\}} \left\{ \frac{k-z}{d-z} T_{(d)} \right\}. \tag{4.6}$$

Theorem 4.6. *The gap between the completion time of PRAC and coded computation using staircase codes is lower bounded by:*

$$\mathbb{E}[T_{SC}] - \mathbb{E}\left[T_{PRAC}\right] \geq \frac{bx - \varepsilon y}{y(x+y)}, \tag{4.7}$$

where $x = \dfrac{n-d^*}{E[\tau_{t,n}]}$, $y = \dfrac{d^*-z}{E[\tau_{t,d^*}]}$ *and* d^* *is the value of* d *that minimizes equation* (4.6).

Proof. The proof of Theorem 4.6 is given in Appendix C.3. □

[7] Note that in addition to n and z, all threshold secret sharing based schemes require a parameter k, $z < k < n$, which is the minimum number of non stragglers that the master has to wait for before decoding $A\mathbf{x}$.

Theorem 4.6 shows that the lower bound on the gap between secure coded computation using Staircase codes and PRAC is in the order of number of row blocks of A. Hence, the gap between secure coded computation using Staircase codes and PRAC is linearly increasing with the number of row blocks of A. Note that, ε, the required overhead by fountain coding used in PRAC, becomes negligible as b increases.

Thus, PRAC outperforms secure coded computation using Staircase codes in heterogeneous systems. The more heterogeneous the workers are, the more improvement is obtained by using PRAC. However, Staircase codes can slightly outperform PRAC in the case where the slowest $n-z$ workers are homogeneous, i.e., have similar compute service times T_i. In this case both algorithms are restricted to the slowest $n-z$ workers (see Lemma 4.4), but PRAC incurs an ε overhead of tasks (due to using Fountain codes) which is not needed for Staircase codes. In particular, from (C.9) and (4.6), when the $n-z$ slowest workers are homogeneous, the task completion time of PRAC and Staircase codes are equal to $\frac{b+\varepsilon}{n-z}\mathbb{E}[\tau_{t,n}]$ and $\frac{b}{n-z}\mathbb{E}[\tau_{t,n}]$, respectively.

4.5 Performance Evaluation

4.5.1 Simulations

In this section, we present simulations run on MATLAB, and compare PRAC with the following baselines: (i) Staircase codes [27], (ii) C3P [110] (which is not secure as it is not designed to be secure), and (iii) Genie C3P (GC3P) that extends C3P by assuming a knowledge of the identity of the eavesdroppers and ignoring them. We note that GC3P serves as a lower bound on private coded computation schemes for heterogeneous systems[8] for the following reason: for a given number of z colluding workers the ideal coded computation scheme knows which workers are eavesdroppers and ignores them to use the remaining workers without need of randomness. If the identity of the colluding workers is unknown, coded computation schemes

[8]If the system is homogeneous Staircase codes outperform GC3P, because pre-allocating tasks to the workers avoids the overhead needed by Fountain codes.

require randomness and become limited to the $(z+1)^{\text{st}}$ fastest worker (Lemma 4.4). GC3P and other coded computation schemes have similar performance if the z colluding workers are the fastest workers. If the z colluding workers are the slowest, then GC3P outperforms any coded computation scheme. Note that our solution PRAC considers the scenario of unknown eavesdroppers. Comparing PRAC with G3CP shows how good PRAC is as compared to the best possible solution for heterogeneous systems. In terms of comparing PRAC to solutions designed for the homogeneous setting, we restrict our attention to Staircase codes which are a class of secret sharing schemes that enjoys a flexibility in the number of workers needed to decode the matrix-vector multiplication. We showed in Chapter 3 that Staircase codes outperform any coded computation scheme that requires a threshold on the number of stragglers.

In our simulations, we model the computation time of each worker $\texttt{W}_i$ by an independent shifted exponential random variable with rate λ_i and shift c_i, i.e., $F(T_i=\tau)=1-\exp(-\lambda_i(\tau-c_i))$. We take $c_i=1/\lambda_i$ and consider three different scenarios for choosing the values of λ_i's for the workers as follows:

- *Scenario 1*: we assign $\lambda_i=3$ for half of the workers, then we assign $\lambda_i=1$ for one quarter of the workers and assign $\lambda_i=9$ for the remaining workers.
- *Scenario 2*: we assign $\lambda_i=1$ for one third of the workers, the second third have $\lambda_i=3$ and the remaining workers have $\lambda_i=9$.
- *Scenario 3*: we draw the λ_i's independently and uniformly at random from the interval $[0.5,9]$.

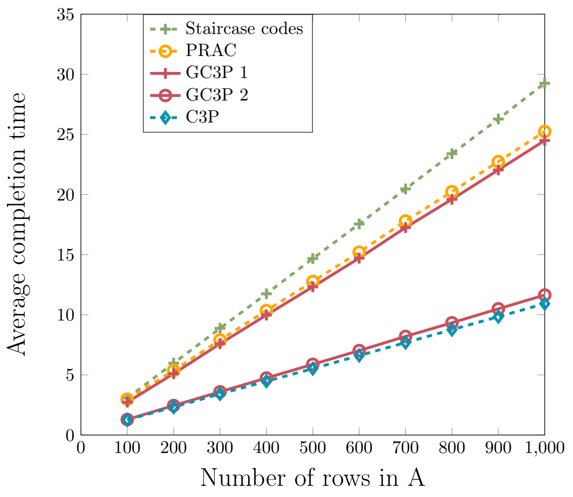
Recall that Staircase codes require an extra parameter k to determine the threshold on the number of stragglers. Therefore, when running Staircase codes, we choose the parameter k that minimizes the task completion time for the desired n and z. We do so by simulating Staircase codes for all possible values of k, $z\leq k\leq n$, and choosing the one with the minimum completion time.

We take $b=m$, i.e., each row block is simply a row of A. The size of each element of A and vector $\mathbf{x}$ are assumed to be 1 Byte (or 8 bits). Therefore, the size of each transmitted packet $P_{t,i}$ is 8ℓ bits. For the simulation results, we assume that the matrix A is a square matrix, i.e., $\ell = m$. We take $m = 1000$, unless explicitly stated otherwise. C_i denotes the average channel capacity of each worker $\texttt{W}_i$ and is selected uniformly from the interval $[10, 20]$ Mbps. The rate of sending a packet to worker $\texttt{W}_i$ is sampled from a Poisson distribution with mean C_i.

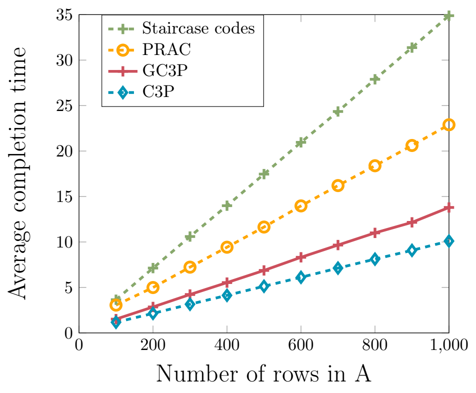
In Figure 4.1 we show the effect of the number of rows m on the completion time at the master. We fix the number of workers to 50 and the number of colluding workers to 13 and plot the completion time for PRAC, C3P, GC3P and Staircase codes. Notice that PRAC and Staircase codes have close completion time in scenario 1 (Figure 4.3a) and this completion time is far from that of C3P. The reason is that in this scenario we pick exactly 13 workers to be fast ($\lambda_i = 9$) and the others to be significantly slower. Since PRAC assigns keys to the fastest z workers, the completion time is dictated by the slow workers. To compare PRAC with Staircase codes notice that the majority of the remaining workers have $\lambda_i = 3$ therefore pre-allocating equal tasks to the workers is close to adaptively allocating the tasks.

In terms of lower bound on PRAC, observe that when the fastest workers are assumed to be adversarial, GC3P and PRAC have very similar task completion time. However, when the slowest workers are assumed to be adversarial the completion of GC3P is very close to C3P and far from PRAC. This observation is in accordance with Lemma 2. In scenarios 2 and 3 we pick the adversarial workers uniformly at random and observe that the completion time of PRAC becomes closer to GC3P when the workers are more heterogeneous. For instance, in scenario 3, GC3P and PRAC have closer performance when the workers' computing times are chosen uniformly at random from the interval $[0.5, 9]$.

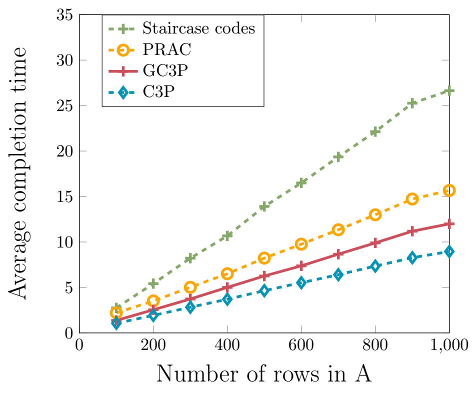
In Figure 4.2, we plot the task completion time as a function of the number of workers n for a fixed number of rows $m = 1000$ and λ_i's assigned according to scenario 1. In Figure 4.2(a), we change the number of workers from 10 to 100 and keep the ratio $z/n = 1/4$ fixed. We notice that with the increase of n the completion time of PRAC becomes closer to GC3P. In Figure 4.2(b), we change the number of workers from 20 to 100 and keep $z = 13$ fixed. We notice that with



(a) Scenario 1 with the fastest 13 workers as eavesdropper for GC3P 1 and the slowest workers as eavesdropper for GC3P 2.



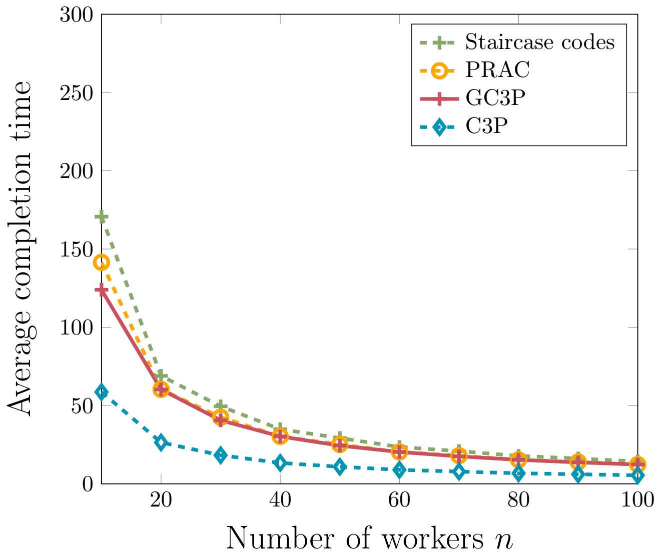
(b) Scenario 2 with 13 workers picked at random to be eavesdroppers.



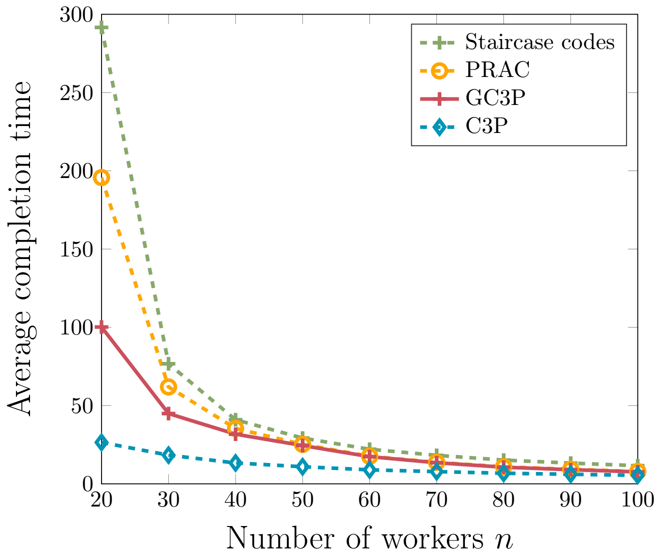
(c) Scenario 3 with 13 workers picked at random to be eavesdroppers.

Figure 4.1: Comparison between PRAC and the baselines Staircase codes, GC3P, and C3P in different scenarios with $n = 50$ workers and $z = 13$ colluding workres for different values of the number of rows m. For each value of m we run 100 experiments and average the results. When the eavesdropper are chosen to be the fastest workers, PRAC has very similar performance to GC3P. When the eavesdroppers are picked randomly, the performance of PRAC becomes closer to this of GC3P when the non adversarial workers are more heterogeneous.

the increase of n, the effect of the eavesdropper is amortized and the completion time of PRAC becomes closer to C3P. In this setting, PRAC always outperforms Staircase codes.



(a) Task completion time as a function of the number of workers with $z = n/4$.



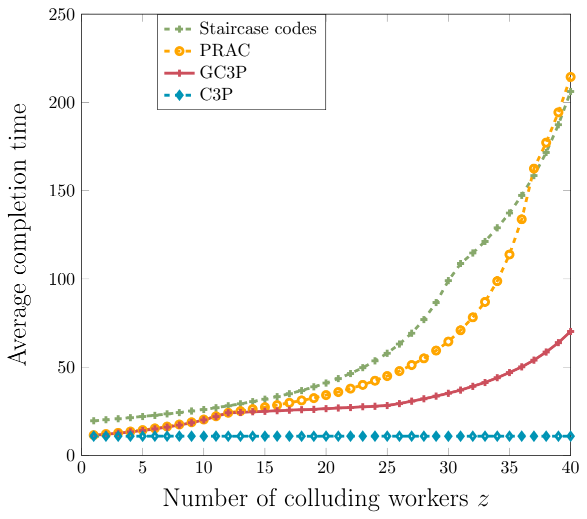
(b) Task completion time as a function of the number of workers with $z = 13$.

Figure 4.2: Comparison between PRAC, Staircase codes and GC3P in scenario 1 for different values of the number workers and number of colluding workers. We fix the number of rows to $m = 1000$. For each value of the x-axis we run 100 experiments and average the results. We observe that the difference between the completion time of PRAC and this of GC3P is large for small values of $n - z$ and decreases with the increase of $n - z$.

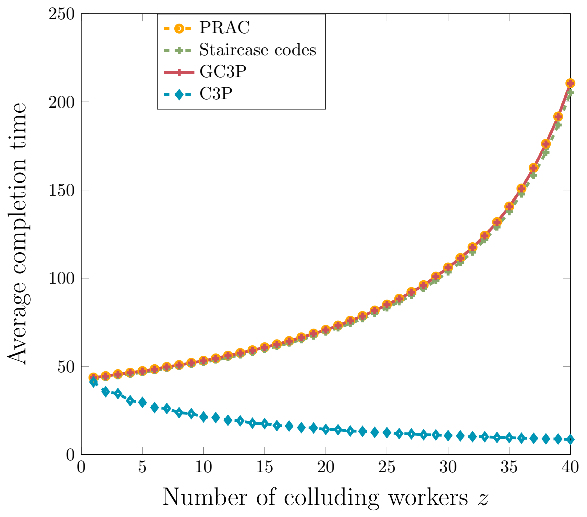
In Figure 4.3, we plot the task completion time as a function of the number of colluding workers. In Figure 4.3(a), we choose the computation time at the workers according to scenario 1. We change z from 1 to 40 and observe that the completion time of PRAC deviates from that of GC3P with the increase of z. More importantly, we observe two inflection points of the average completion time of PRAC at $z = 13$ and $z = 37$. Those inflection points are due to the fact that we have 12 fast workers ($\lambda = 9$) and 25 workers with medium speed ($\lambda = 3$) in the system. For $z > 36$, the completion time of Staircase codes becomes less than that of PRAC because the 14 slowest workers are homogeneous. Therefore, pre-allocating the tasks is better than using Fountain codes and paying for the overhead of computations. To show that Staircase codes always outperforms PRAC when the slowest $n - z$ workers are homogeneous, we run a simulation in which we divide the workers into three clusters. The first cluster consists of $\lfloor z/2 \rfloor$ fast workers ($\lambda = 9$), the second consists of $\lfloor z/2 \rfloor + 1$ workers that are regular ($\lambda = 3$) and the remaining $n - z$ workers are slow ($\lambda = 1$). In Figure 4.3(b) we fix n to 50 and change z from 1 to 40. We observe that Staircase codes always outperform PRAC in this setting. In contrast to non secure C3P, Staircase codes and PRAC are always restricted to the slowest $n - z$ workers and cannot leverage the increase of the number of fast workers. For GC3P, we assume that the fastest workers are eavesdroppers. We note that as expected from Lemma 4.4, when the fastest workers are assumed to be eavesdroppers the performance of GC3P and PRAC becomes very close.

4.5.2 Implementation on Android devices

Setup. The master device is a Nexus 5 Android-based smartphone running 6.0.1. The worker devices are Nexus 6Ps running Android 8.1.0. The master device connects to worker devices via Wi-Fi Direct links and the master is the group owner of Wi-Fi Direct group. The master device is required to complete one matrix multiplication ($y = Ax$) where A is of dimensions 60×10000 and x is a 10000×1 vector. We also take $m = b$ i.e., each packet is a row of A. We introduced an artificial delay at the workers following an exponential distribution. The introduced delays serves to emulate applications running in the background of the devices. A



(a) Task completion time as a function of the number of colluding workers for $n = 50$. Computation time of the workers are chosen according to scenario 1.



(b) Task completion time for $n = 50$ workers and variable z. Computation times of the workers are chosen such that the $n - z$ slowest workers are homogeneous.

Figure 4.3: Comparison between PRAC and Staircase codes average completion time as a function of number of colluding workers z. We fix the number of rows to $m = 1000$ and design the system in a way that $n - z$ workers are significantly slower than the other z workers. We observe that PRAC outperforms Staircase codes except when the $n - z$ slowest workers are homogeneous. All secure codes are affected by the increase of number of colluding helpers because their runtime is restricted to the slowest $n - z$ workers. In contrast, C3P leverages the increase of the number of fast workers.

worker device sends the result to the master after it is done calculating and the introduced delay has passed. Furthermore, we assume that $z = 1$ i.e., there is one unknown worker that is

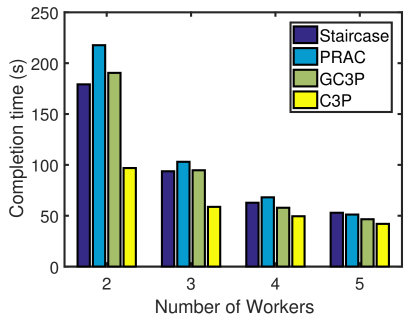


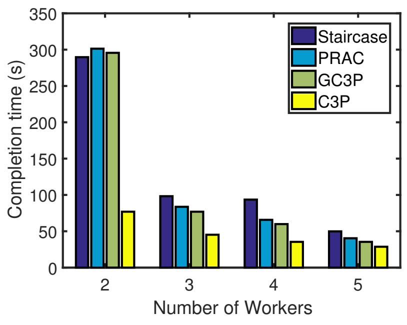
Figure 4.4: Completion time as function of the number of workers in homogeneous setup.

adversarial among all the workers. The experiments are conducted in a lab environment where there are other Wi-Fi networks operating in the background.

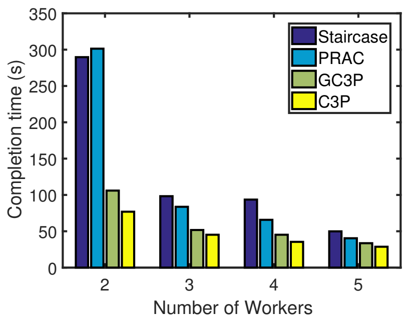
Baselines. Our PRAC algorithm is compared to three baseline algorithms: (i) Staircase codes that preallocate the tasks based on n, the number of workers, k, the minimum number of workers required to reconstruct the information, and z, the number of colluding workers; (ii) GC3P in which we assume the adversarial worker is known and excluded during the task allocation; (iii) Non secure C3P in which the security problem is ignored and the master device will utilize every resource without randomness. In this setup we run C3P on $n - z$ workers.

Results. We present a representative sample of our results and refer interested readers to [] form more results and explanations. Figure 4.4 presents the task completion time with increasing number of workers for the homogeneous setup, i.e., when all the workers have similar computation times. Computation delay for each packet follows an exponential distribution with mean $\mu = 1/\lambda = 3$ seconds in all workers. C3P performs the best in terms of completion time, but C3P does not provide any privacy guarantees. PRAC outperforms Staircase codes when the number of workers is 5. The reason is that PRAC performs better than Staircase codes in heterogeneous setup, and when the number of workers increases, the system becomes a bit more heterogeneous. GC3P significantly outperforms PRAC in terms of completion time. Yet, it requires a prior knowledge of which worker is adversarial, which is often not available in real world scenarios.

We note that in all our experiments when $n - z$ slowest workers are homogeneous Staircase



(a) We assume a fast worker is adversarial for GC3P.



(b) We assume a slow worker is adversarial for GC3P.

Figure 4.5: Completion time as function of the number of workers in heterogeneous setup.

codes outperform GC3P and PRAC. This happens because pre-allocating the tasks to the workers avoids the overhead of sub-tasks required by Fountain codes and utilizes all the workers to their fullest capacity.

4.6 Related work

Mobile cloud computing is a rapidly growing field with the aim of providing better experience of quality and extensive computing resources to mobile devices [121, 122]. The main solution to mobile computing is to offload tasks to the cloud or to neighboring devices by exploiting connectivity of the devices. With task offloading come several challenges such as heterogeneity of the devices, time varying communication channels and energy efficiency, see e.g., [123–126]. We refer interested reader to [110] and references within for a detailed literature on edge computing and mobile cloud computing.

The problem of stragglers in distributed systems is initially studied by the distributed computing community, see e.g., [5,76,82,93]. Research interest in using coding theoretical techniques for straggler mitigation in distributed content download and distributed computing is rapidly growing. The early body of work focused on content download, see e.g., [26, 94–96, 127]. Using codes for straggler mitigation in distributed computing started in [6] where the authors proposed the use of MDS codes for distributed linear machine learning algorithms in homogeneous workers setting.

Following the work of [6], coding schemes for straggler mitigation in distributed matrix-matrix multiplication, coded computation and machine learning algorithms are introduced and the fundamental limits between the computation load and the communication cost are studied, see e.g., [97, 128] and references within for matrix-matrix multiplication, see [6, 10–13, 15, 100, 101, 111, 129–133] for machine learning algorithms and [7, 98, 99, 134] and references within for other topics.

Codes for privacy and straggler mitigation in distributed computation are first introduced in [27] where the authors consider a homogeneous setting and focus on matrix-vector multiplication. Beyond matrix-vector multiplication, the problem of private distributed matrix-matrix multiplication and private polynomial computation with straggler tolerance is studied [19–21, 102, 103, 135]. The former works are designed for the homogeneous static setting in which the master has a prior knowledge on the computation capacities of the workers and pre-assigns the sub-tasks equally to them. In addition, the master sets a threshold on the number of stragglers that it can tolerate throughout the whole process. In contrast, PRAC is designed for the heterogeneous dynamic setting in which workers have different computation capacities that can change over time. PRAC assigns the sub-tasks to the workers in an adaptive manner based on the estimated computation capacity of each worker. Furthermore, PRAC can tolerate a varying number of stragglers as it uses an underlying rateless code, which gives the master a higher flexibility in adaptively assigning the sub-tasks to the workers. Those properties of PRAC allow a better use of the workers over the whole process. On the other hand, PRAC is restricted to matrix-vector multiplication. Although coded computation is designed for linear operations, there is a recent effort to apply coded computation for non-linear operations. For example, [104] applied coded computation to logistic regression, and the framework of Gradient coding started in [10] generalizes to any gradient-descent algorithm. Our work is complementary with these works. For example, our work can be directly used as complementary to [104] to provide privacy and adaptive task offloading to logistic regression.

Secure multi-party communication (SMPC) [4] can be related to our work as follows. The setting of secure multi-party computation schemes assumes the presence of several parties (masters in our terminology) who want to compute a function of all the data owned by the different parties without revealing any information about the individual data of each party. This setting is a generalized version of the master/worker setting that we consider. More precisely, an SMPC scheme reduces to our Master/worker setting if we assume that only one party owns data and the others have no data to include in the function to be computed. SMPC schemes use threshold secret sharing schemes, therefore they restrict the master to a fixed number of stragglers. Thus, showing that PRAC outperforms Staircase codes (which are the best known family of threshold secret sharing schemes) implies that PRAC outperform the use of SMPC schemes that are reduced to this setting. Works on privacy-preserving machine learning algorithms are also related to our work. However, the privacy constraint in this line of work is computational privacy and the proposed solutions do not take stragglers into account, see e.g., [106, 107, 109].

We restrict the scope of this paper to eavesdropping attacks, which are important on their own merit. Privacy and security can be achieved by using Maximum Distance Separable (MDS)-like codes which restrict the master to a fixed maximum number of stragglers [102, 103]. Our solution on the other hand addresses the privacy problem in an adaptive coded computation setup without such a restriction. In this setup, security cannot be addressed by expanding the results of [102, 103]. In fact, we developed a secure adaptive coded computation mechanism in our recent paper [136] against Byzantine attacks. The private and secure adaptive coded computation obtained by combining this paper and [136] is out of scope of this paper.

4.7 Conclusion

The focus of this Chapter is to develop an adaptive private coded computation scheme that allows the master to adapt to the varying resources at the workers and mitigate the effect of stragglers. Focusing on eavesdropping attacks, we designed a private and rateless adaptive coded computation (PRAC) mechanism considering (i) the privacy requirements of IoT applications and devices, and (ii) the heterogeneous and time-varying resources of the workers. Our proposed PRAC model can provide adequate privacy and latency guarantees to support real-time computation at the edge. We showed through analysis and MATLAB simulations that PRAC outperforms known private coded computtion methods when resources are heterogeneous.

Chapter 5

Stochastic Gradient Coding for Straggler Mitigation in Distributed Learning

We go beyond coded matrix-vector multiplications and consider coded computation for general distributed gradient descent algorithms in the presence of stragglers. In this chapter, we drop the privacy constraints on the master's data. Recent work on *gradient coding* and *approximate gradient coding* have shown how to add redundancy in distributed gradient descent to guarantee convergence even if some workers are *stragglers*. We propose an approximate gradient coding scheme called *Stochastic Gradient Coding* (SGC), which works when the stragglers are random. SGC distributes data points redundantly to workers according to a pair-wise balanced design, and then simply ignores the stragglers. We prove that the convergence rate of SGC mirrors that of batched Stochastic Gradient Descent (SGD) for the ℓ_2 loss function, and show how the convergence rate can improve with the redundancy. We also provide bounds for more general convex loss functions. We show empirically that SGC requires a small amount of redundancy to handle a large number of stragglers and that it can outperform existing approximate gradient codes when the number of stragglers is large.

5.1 Introduction

We consider the same setting where a master wants to run a gradient-descent-like algorithm to solve an optimization problem distributed across several workers. Let $B \in \mathbb{R}^{m\times\ell}$ be the data matrix and let $\mathbf{b}_i \in \mathbb{R}^\ell$ denote the i^{th} row of B. Let $\mathbf{y} \in \mathbb{R}^m$ be a vector of labels, so $\mathbf{b}_i$ has label y_i. Define $A \triangleq [B|\mathbf{y}]$ to be the concatenation[1] of B and $\mathbf{y}$. The master wants to find a vector $\mathbf{x}^\star \in \mathbb{R}^\ell$ that best represents the data B as a function of the labels $\mathbf{y}$. That is, the goal is to iteratively solve an optimization problem

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \mathcal{L}(A, \mathbf{x}), \tag{5.1}$$

for a given loss function $\mathcal{L}$, by simulating or approximating an update rule of the form

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t \nabla \mathcal{L}(A, \mathbf{x}_t). \tag{5.2}$$

Many natural loss functions $\mathcal{L}(A, \mathbf{x})$ can be written as the sum over individual rows $\mathbf{a}_i$ of A, i.e.,

$$\mathcal{L}(A, \mathbf{x}) = \sum_{i=1}^{m} \mathcal{L}(\mathbf{a}_i, \mathbf{x}), \tag{5.3}$$

such loss functions lend themselves naturally to distributed algorithms. In a distributed setting, the master partitions the data matrix A into rows $\mathbf{a}_i$ which are distributed between the workers. Each worker returns some linear combination(s) of the gradients $\nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x}_t)$ that it can compute, and the master aggregates these together to compute or approximate the update step (5.2).

We focus on the setting where some of the workers may be *stragglers,* i.e., slow or unresponsive. This setting has been studied before in the systems community [5, 39, 79, 137], and

[1] In contrast to the remaining of the dissertation, in this chapter the master wants to send the vector $\mathbf{y}$ to the workers. Therefore, we adapt the notation so that the matrix A is the one sent to the workers.

recently in the coding theory community [6, 10, 12]. A typical approach is to introduce some redundancy: for example, the same piece of data $\mathbf{a}_i$ might be held by several workers. There are several things that one might care about in such a scheme and in this paper we focus on the following four desiderata:

(A) **Convergence speed.** We would like the error $\|\mathbf{x}_t - \mathbf{x}^\star\|_2$ to shrink as quickly as possible.

(B) **Redundancy.** We would like to minimize the amount of storage and computation overhead needed between the workers.

(C) **Communication.** We would like to minimize the amount of communication between the master and the workers.

(D) **Flexibility.** In practice, there is a great deal of variability in the number of stragglers over time. We would like an algorithm that degrades gracefully if more stragglers than expected occur.

Exact gradient coding for worst-case stragglers. Much existing work has focused on simulating gradient descent *exactly,* even in the presence of worst-case stragglers, for example [6, 10, 12, 13]. In that model, at each round an arbitrary set of s workers (for a fixed s) may not respond to the master. The goal is for the master to obtain the same update $\mathbf{x}_t$ at round t that gradient descent would obtain. For this to happen, the master should be able to obtain an exact value of the gradient $\nabla\mathcal{L}(A, \mathbf{x}_t)$. This has given rise to (exact) *gradient coding* [10], which focuses on optimizing desiderata (A) and (C) above. However, these schemes (and necessarily, any scheme in this model) do not do so well on (B) and (D). First, it is not hard to see that in the presence of s worst-case stragglers, it is necessary for any $n - s$ workers to be able to recover all of the data, which necessitates a certain amount of overhead. Namely, every data vector should be replicated on $s + 1$ different workers. Second, the gradient coding schemes for example in [10, 12] are brittle in the sense that they work perfectly for s failures, but cannot handle more than s stragglers.

No coding at all for random stragglers. On the other hand, there has also been work on *approximately* simulating gradient descent. One approach (similar to the one in [39]) is to assume that the stragglers are random, rather than worst-case, and not employ any redundancy at all. Thus, the master obtains an approximate update (5.2) instead of an exact one by computing the sum in (5.3) without the responses of the stragglers. (We will later refer to this algorithm as "Ignore–Stragglers–SGD.") If the stragglers are independent at each round, this algorithm is a close approximation to Batch–SGD, see e.g. [138–141], and performs in about the same way. However, for convex loss functions it is well known that, while Batch–SGD does converge to $\mathbf{x}^\star$, the convergence is not as fast as that of classical gradient descent [36–38]. Thus, this approach maintains the good communication cost (C) of the coded approaches by requiring each worker to send one linear combination of the gradients to the master, and improves on (B) and (D), but sacrifices (A), the convergence rate.

Approximate gradient coding: adding redundancy to approximate the gradient. A line of work known as *approximate gradient coding* [13–17, 40, 41] introduces redundancy in order to speed up the convergence rate of such an approximate scheme. This line of work studies the data redundancy d (that is, the number of times each row $\mathbf{a}_i$ of the data matrix A is replicated) needed to tolerate s stragglers and allow the master to compute an approximation of the gradient if more than s workers are stragglers [13, 14, 16, 17]. In [15] a variant of this idea is studied; in that work the data is encoded using LDPC code rather than being duplicated. In approximate gradient coding, the master is required to compute the exact gradient with high

Approach	Stragglers?	(A)	(B)	(C)	(D)
Full gradient descent	No	✓	✓	X	N/A
Exact gradient coding	Worst-case stragglers	✓	X	✓	X
No coding	Random stragglers	X	✓	✓	✓
Appx. gradient coding	Random* stragglers	✓	✓	✓	✓

Table 5.1: High-level overview of the trade-offs offered by distributed gradient-descent various models. In this work, we focus on provable guarantees on the trade-offs between (A) and (B) in the approximate gradient coding model, and demonstrate that our scheme simultaneously achieves (A),(B),(C),(D).

*We note that there are works in approximate gradient coding which study worst-case stragglers, but we focus on the model with random stragglers. See Remark 5.1.

probability if fewer than s workers are stragglers. If more than s workers are stragglers, the distance between the computed gradient at the master and the true gradient can be made small if the redundancy factor is poly-logarithmic in the number of workers. So far, this line of work has mostly focused on desiderata (B), (C) and (D), and most works have not directly analyzed the convergence time (A). Two exceptions are [15] and [17], which we discuss more below.

These strengths and weaknesses are summarized at a high level in Table 5.1.

We introduce an approximate gradient coding scheme called *Stochastic Gradient Coding* (SGC) which works in the random straggler model and which does well simultaneously on desiderata (A)-(D). We analyze the convergence rate of SGC, and we present experimental work which demonstrates that SGC outperforms the most recently proposed schemes [14, 17] when p (the fraction of workers that the master will ignore in each iteration) is relatively large.

Remark 5.1 (Motivation for the random straggler model, and for large p.). A model of random stragglers has been studied before (e.g., [13–17, 40, 142]), and is motivated as an easy-to-study model that captures non-persistent stragglers. However, one might also work in a "random stragglers" model even if all of the workers are fast: just like how SGD samples fewer points to save time and computation, so the master might sample random workers to save bandwidth and computation.

This second setting further motivates the case when p might be relatively large, which is the setting that we focus on in this work.

5.1.1 Contributions

We consider an approach that we call *Stochastic Gradient Coding* (SGC). The coding idea—which is similar to previous approaches in approximate gradient coding [14]—is simple: the master distributes data to the workers with a small amount of repetition according to a *pair-wise balanced scheme* (which we will define below); a data point $\mathbf{a}_i$ is replicated d_i times, and d_i can vary from data point to data point. Below, the redundancy parameter d refers to the average of the d_i's. Once the data is distributed, the algorithm proceeds similarly to the Ignore–Stragglers–SGD algorithm described above: workers compute gradients on their data and return a linear combination, and the master aggregates all of the linear combinations it receives to do an update step.

Remark 5.2 (The role of redundancy in SGC and in approximate gradient coding with random stragglers.). Since our scheme is replication-based, the reader may wonder why we use the word "coding." Here, we are using it in the same way as is standard to describe the many replication-based schemes in the gradient coding literature, for example [10, 13, 14, 17].[2] Since each worker responds with only a single vector (rather than all of the partial gradients it can compute), and a worker (with all its data) straggles as a unit, it matters how the data is distributed between the workers. We will see that this matters in our experiments in Section 5.6, where we compare the SGC method of distributing data to the different data distribution methods of [17].

One point of our work is to understand to what extent adding redundancy can speed up the convergence of SGD. To this end, we will also compare SGC with the "Ignore–Stragglers–SGD" algorithm alluded to above, where there is no replication of the data and the master simply ignores slow workers when estimating the gradient.

One contribution of this work is to provide a rigorous convergence analysis of SGC. We show that SGC with only a small amount of redundancy d is able to regain the benefit of (A) from the (exact) coded approaches, while still preserving the benefits of (B), (C), (D) that the "Ignore–Stragglers–SGD" approach sketched above does. A second contribution is extensive experimental evidence which suggests that for the same small redundancy factor d SGC outperforms other schemes when there are many stragglers.

More precisely, our contributions are as follows (all in the stochastic straggler model):

- In the special case of the ℓ_2 loss function, we show that SGC with redundancy factor $d > 1$, can obtain error bounds where $\|\mathbf{x}^\star - \mathbf{x}_t\|_2$ decreases at first exponentially and then proportionally to $\frac{1}{td}$. This mirrors existing results on SGD (which corresponds to the case

[2] We note also that our replication-based data distribution is similar to a *Fractional Repetition* (FR) code [143].

$d = 1$), and quantifies the trade-off between replication and error. This is made formal in Theorem 5.6.

- For more general loss functions, we show that SGC has at least the same convergence rate as Ignore–Stragglers–SGD, and we give some theoretical evidence that the error $\|\mathbf{x}^\star - \mathbf{x}_t\|$ may decrease as d increases. This is made formal in Theorem 5.9.

- We provide numerical simulations comparing SGC to gradient descent, Ignore–Stragglers–SGD and a few other versions of SGD, and other approximate gradient coding methods. Our simulations show that indeed SGC improves the accuracy of Ignore–Stragglers–SGD, with far less redundancy than would be required to implement exact gradient descent using coding. In addition, we compare SGC to other approximate gradient methods existing in the literature and show that SGC outperforms the existing methods when the probability of workers being stragglers is high.

5.1.2 Relationship to previous work on approximate gradient coding

We provide a more detailed description of previous work in Section 5.8, but first we briefly mention some of the main differences between our work and existing work on approximate gradient coding [13–17, 40].

First, we note that our SGC scheme is quite similar to Bernoulli Gradient Coding (BGC) studied in [14], where the data is distributed uniformly at random to d workers. One difference between our work and that work is that we allow for the redundancy of different data points $\mathbf{a}_i$ to vary for different i; we will see that for the ℓ_2 loss function it makes sense to choose d_i based on $\|\mathbf{b}_i\|_2$. A second difference between our work and [14] is that [14] does not provide a complete convergence analysis. The works [13, 16, 40] also study schemes similar in flavor to SGC, but these works also do not provide complete convergence analyses.

The works of [15, 17] do provide convergence analyses, although for schemes that are quite different from SGC. More precisely, [15] studies a scheme with LDPC coding, rather than repetition. The work of [17] studies a scheme based on Fractional Repetition (FR) codes, which was proposed in [14]. However, the FR codes studied by [17] results in a very different data distribution scheme than the one we study. Their scheme partitions the data and the workers into different blocks and every worker in a block receives all of the data from the corresponding block.

Additionally, we obtain slightly different error guarantees than the analyses of [15,17]. More precisely, the analysis of [17] proves a bound where the error decreases exponentially in T (the number of iterations of the algorithm) until some noise floor is hit. The analysis of [15] studies the special case of the ℓ_2 loss function, and shows that the error decays like $\mathcal{O}(1/\sqrt{T})$. In contrast, for SGC and for the special case of the ℓ_2 loss function, we show that the error decays exponentially in T at first and then switches to decaying like $\mathcal{O}(1/T)$; this mirrors existing results for SGD for the ℓ_2 loss function. We give a more general result that holds for general convex loss functions and show that the error decays as $\mathcal{O}(1/T)$.

Finally, we provide empirical results which suggest that our scheme can outperform existing gradient coding schemes (in particular, the FR-based approach of [14,17] and BGC [14]) in some parameter regimes. We do not compare our scheme empirically to that of [15, 40, 41] because they requires more work on the master's end (to encode and decode) and are thus not directly comparable to our work.

5.1.3 Organization

We give a more precise definition of our set-up in Section 5.2. We describe the SGC algorithm in Section 5.3. In Section 5.4, we give a more detailed overview of both our theoretical and empirical results, which are fleshed out in Sections 5.5 and 5.6 respectively. We provide more detail on related work in Section 5.8.

5.2 Setup

5.2.1 Probabilistic model of stragglers

In this paper, we adopt a probabilistic model of stragglers. More precisely, we assume that at every iteration each worker may be a straggler with some probability p, and this is independent between workers and between iterations. Our probabilistic model is similar to the model in [13–17, 40] and is in contrast to the worst-case model assumed by much of the literature on coded computation. (See Remark 5.1). In our numerical simulations, we relax the assumption of independence and show that similar results hold when the identities of the stragglers are somewhat persistent from round to round and change only after a fixed number of iterations.

5.2.2 Computational model

Our computational model has two stages, a distribution stage and a computation stage.

In the *distribution stage,* the master encodes the data using unequal data repetition code. More precisely, the master can decide to send each row $\mathbf{a}_i$ of A to d_i different workers. We refer to the parameter $d = \frac{1}{m}\sum_{i=1}^{m} d_i$ as the *average redundancy* of the scheme.

The *computation stage* is made up of rounds, each of which contains two repeating steps. In the first step, the master does some local computation and then sends a message to each worker. In the second step, each worker does some local computation and tries to send a message back to the master; however, with probability p the message may not reach the master. Then the round is over and the master repeats the first step to begin the next round. We refer to the total amount of communication per round as the *communication* of the scheme. We allow each worker to send only one message to the master to reduce the communication.

5.3 Stochastic Gradient Coding

In this section, we describe our solution, which we call *Stochastic Gradient Coding* (SGC). The idea behind SGC is extremely simple. It is very much like the Ignore–Stragglers–SGD

Symbol	Meaning	Symbol	Meaning
n	number of workers	A	data matrix of the master
$\mathbf{a}_i$	i^{th} row of A	$\mathbf{y}$	label vector of dimension $m \times 1$
$\mathbf{b}_i$	i^{th} row of B	B	the concatenation of A and $\mathbf{y}$
m	number of rows in A	$\mathtt{W}_j$	worker j
ℓ	number of columns in A	$\mathcal{S}_j$	indices of rows of A given to $\mathtt{W}_j$
t	iteration number	T	total number of iterations
$\mathbf{x}^\star$	desired vector of dimension $\ell \times 1$	$\mathbf{x}_t$	value of $\mathbf{x}$ at iteration t
d_i	replication factor of $\mathbf{a}_i$	$\mathcal{L}(A, \mathbf{x}_t)$	loss function evaluated at A and $\mathbf{x}$
γ_t	step size at iteration t	$\nabla\mathcal{L}(A, \mathbf{x}_t)$	gradient of the loss function evaluated at A and $\mathbf{x}$
d	average redundancy $d \triangleq \sum_{i=1}^{m} d_i/m$	p	probability of worker being straggler

Table 5.2: Summary of notations for this Chapter.

algorithm described above, except we introduce a small amount of redundancy. We describe the distribution stage and the computation stage of our algorithm below. Our scheme has parameters $d_1, \dots, d_m$, which control the redundancy of each row, and a parameter γ_t which controls the step size. We will see in the theoretical and numerical analyses how to set these parameters.

In our analysis, we focus on *pair-wise balanced schemes:*

Definition 5.3. We say that a distribution scheme that sends $\mathbf{a}_i$ to d_i different workers is *pair-wise balanced* if for all $i \neq i'$, the number of workers that receives $\mathbf{a}_i$ and $\mathbf{a}_{i'}$ is $\frac{d_i d_{i'}}{n}$.

Notice that with a completely random distribution scheme, the expected number of workers who receive both $\mathbf{a}_i$ and $\mathbf{a}_{i'}$ for $i \neq i'$ is equal to $\frac{d_i d_{i'}}{n}$. In our analysis, it is convenient to deal with schemes that are exactly pair-wise balanced. However, for small d_i it is clear that no such schemes exist (indeed, we may have $\frac{d_i d_{i'}}{n} < 1$). In our simulations, we choose a uniformly random scheme[3] which seems to work well (see Section 5.6). We believe that our analysis should extend to a random assignment as well, although for simplicity we focus on pair-wise balanced schemes in our theoretical results.

The way SGC works is as follows:

- **Distribution Stage**. The master creates d_i copies of each row $\mathbf{a}_i$, $i = 1, \dots, m$, and

[3]In our simulations, we assign rows to d_i workers uniformly at random, which approximates a pair-wise balanced scheme. Similarly, the BGC construction of [14] approximates a pair-wise balanced scheme where each row is assigned to d workers uniformly at random, i.e., $d_i = d$ for all $i \in [m]$.

sends them to d_i distinct workers according to a pair-wise balanced scheme. We denote by $\mathcal{S}_j$, $j = 1, \ldots, n$, the set of indices of the data vectors given to worker $\mathtt{W}_j$, i.e., $\mathcal{S}_j = \{i; \mathbf{a}_i \text{ is given to } \mathtt{W}_j\}$.

- **Computation Stage.** At each iteration t, the master sends $\mathbf{x}_t$ to all the workers. Each worker $\mathtt{W}_j$ computes

$$f_j(\mathbf{x}_t) \triangleq \gamma_t \sum_{i \in \mathcal{S}_j} \frac{1}{d_i(1-p)} \nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x}_t) \tag{5.4}$$

and sends the result to the master. The master aggregates all the received answers from non straggler workers, sums them and updates $\mathbf{x}$ as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t \sum_{j=1}^{n} \sum_{i=1}^{m} \frac{\mathcal{I}_i^j}{d_i(1-p)} \nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x}_t),$$

where $\mathcal{I}_i^j$ is the indicator function for worker j being non straggler and having obtained point $\mathbf{a}_i$ during the data distribution, i.e.,

$$\mathcal{I}_i^j = \begin{cases} 1 & \text{if worker } j \text{ is non straggler and has point } \mathbf{a}_i, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\mathcal{I}_i^j$ depends on the iteration t, however we drop t from the notation for notational convenience since the value of t will be clear from the context.

For use below, we define

$$\hat{\mathbf{g}}_t \triangleq \sum_{j=1}^{n} \sum_{i=1}^{m} \frac{\mathcal{I}_i^j}{d_i(1-p)} \nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x}_t). \tag{5.5}$$

We call $\hat{\mathbf{g}}_t$ the estimate of the gradient at iteration t which estimates the exact gradient of the loss function in (5.3),

$$\mathbf{g}_t \triangleq \sum_{i=1}^{m} \nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x}_t).$$

5.4 Summary of our Main Results

In this section, we summarize both our theoretical and numerical results.

5.4.1 Theoretical results

Our main theoretical contributions are to derive results for SGC that mirror known results for SGD and Batch–SGD. There are two important differences between our results and those for Batch–SGD.

1. First, one of our goals is to show how the error $\|\mathbf{x}^\star - \mathbf{x}_t\|_2^2$ depends on the redundancy parameter d; we show that it is roughly like $1/d$. This explains why SGC can work much better than Ignore–Stragglers–SGD (say, so that $\|\mathbf{x}_t - \mathbf{x}^\star\|_2^2$ is half as large), even with relatively low redundancy (say, $d = 2$). In Batch–SGD we always have $d = 1$.

2. Second, it is nontrivial to adapt existing results for Batch–SGD to our setting. The reason is that the batches are not uniform in our setting; rather, they depend on the way that the data is distributed. We note that this is true even if the data is distributed randomly to begin with: in that case it is true that the marginals of the batches are uniformly random (that is, in each round the set of gradients that the master receives is a uniformly random subset of all of them) but because the randomness from the initial distribution is fixed throughout the computation, if we view it this way then the batches are no longer independent. The main technical challenge in our analysis (in particular, the proof of Theorem 5.4 below) is to deal with this issue.[4]

We adapt existing result from the SGD literature to prove a tighter bound that holds for arbitrary convex loss functions. And we derive a stronger convergence guarantee for the ℓ_2 loss function.

Special case: ℓ_2 loss function. We begin with a result which is specialized for the ℓ_2 loss

[4]We note that this is not an issue for our proof of Theorem 5.5, since we are able to adapt existing results that depend only on the mean and variance of the gradient estimates.

function. This result is of a similar flavor as the results of [144–146] on SGD and the randomized Kaczmarz algorithm.[5] Those works show that the speed of convergence is exponential to begin with, and then begins to decay polynomially like $1/t$ once an unavoidable limit is reached. In this work, we show an analogous result for the ℓ_2 loss function. In this case we show that the convergence is exponential to begin with, until the noise is on the order of ℓ_2 normal of the residual $\mathbf{r} \triangleq B\mathbf{x}^\star - \mathbf{y}$, and then it begins to decay polynomially like $1/(dt)$.

Thus, our analysis generalizes the case when $d = 1$ (aka, Ignore–Stragglers–SGD), and we see that as the repetition factor d increases, the error of SGC decreases. We state our main theorem informally below, and we state the formal version in Section 5.5. Throughout the paper we use the superscript T to denote the transpose of a matrix.

Theorem 5.4 (Informal; see Theorem 5.6 for a formal version)**.** *Consider an SGC algorithm run on a matrix $A \triangleq [B|\mathbf{y}]$ of dimension $m \times (\ell+1)$ distributed to n workers. Suppose that the distribution scheme is pairwise balanced, and that each row $\mathbf{a}_i$ of $A = [B|\mathbf{y}]$ is sent to d_i different workers, where d_i is chosen proportional to $\|\mathbf{b}_i\|_2^2$.*

Suppose that n is sufficiently large and that

$$d = \frac{1}{m}\sum_{i=1}^{m} d_i \geq 8\left(\frac{p}{1-p}\right).$$

Choose an error tolerance $\varepsilon > 0$. Then, it is possible to choose a step size γ_t at each step t so that the following guarantee holds on the iterates $\mathbf{x}_T$ of SGC, for $T \geq 2\log(1/\varepsilon^2)$:

$$\mathbb{E}\left[\|\mathbf{x}_T - \mathbf{x}^\star\|_2^2\right] \leq \varepsilon^2\|\mathbf{x}_0 - \mathbf{x}^\star\|_2^2 + \frac{1}{d\cdot T}\cdot\left(\log^2(1/\varepsilon)\frac{p}{1-p}\right)\cdot\|\tilde{\mathbf{r}}\|^2,$$

where $\tilde{\mathbf{r}} = (B\mathbf{x}^\star - \mathbf{y})/\|B^T X\|_2$.

That is, if the residual $\tilde{\mathbf{r}}$ is very tiny, so that the second term is smaller than the first, then the algorithm reaches accuracy ε in roughly $\log(1/\varepsilon)$ steps. However, if $\tilde{\mathbf{r}}$ is larger, then the convergence becomes polynomial, matching what we expect from SGD. In this second case, the

[5]We note that [145] also holds for more general loss functions.

difference is that the replication factor d appears in the denominator, so that when d is larger, the error is smaller, explaining why replication helps. Notice that if p is constant, we expect good performance when $d = O(1)$. In contrast, to exactly simulate gradient descent via coding would require $d = \Omega(n)$.

The main difficulty in proving Theorem 5.6 (the formal version of Theorem 5.4) is that because the data distribution is fixed ahead of time, the "batches" that the master acquires in each round are not uniformly random, but rather come from some distribution determined by the data distribution.

Beyond ℓ_2 loss function. Our result above is limited in that it only applies to the ℓ_2 loss function. We believe that the analysis of Theorem 5.6 should apply to general loss functions, but for now we observe that in fact a convergence rate of $1/t$ does follow for SGC from a result of [36].

In that work, the authors give a general analysis of stochastic gradient descent, which works as long as (in our language) the master is computing an unbiased estimator of the gradient. The convergence speed of the algorithm then depends on the variance of this estimate. This result applies in our setting:

Theorem 5.5 (Informal; see Theorem 5.9 for a formal version)**.** *Suppose that SGC is run on a matrix $A \triangleq [B|\mathbf{y}]$ of dimension $m \times (\ell+1)$ distributed to n workers. Suppose that the distribution scheme is pairwise balanced, and that each row $\mathbf{a}_i$ of A is sent to d_i different workers, $d_i \leq n$. Consider a version of the optimization problem in (5.1) where $\mathbf{x}$ is constrained to a convex set $\mathcal{W}$. Under some mild assumptions on the loss function $\mathcal{L}$ and assuming there exists a constant C such that*

$$\|\nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x})\|_2^2 \leq C^2$$

for all $i \in [n]$ and for all $\mathbf{x} \in \mathcal{W}$, then there is a way to choose the step size γ_t at each step t so that the error after T iterations is bounded by

$$\mathbb{E}\left[\|\mathbf{x}_T - \mathbf{x}^\star\|_2^2\right] \leq \mathcal{O}(1/T).$$

The proof of Theorem 5.5 (given Lemma 1 in [36]) boils down to showing that our gradient estimator $\hat{\mathbf{g}}_t$ is an unbiased estimator of the true gradient and that $\mathbb{E}\left[\|\hat{\mathbf{g}}_t\|_2^2\right]$ is bounded for all t, which we do in Section 5.7.

We give more precise statements of these theorems in Section 5.5, and prove them in Section 5.7.

5.4.2 Numerical simulations

We run extensive simulations on synthetic data A of dimension 1000×100 generated from a Gaussian distribution. We compare SGC to four other algorithms detailed in Section 5.6 and show that SGC outperforms all other algorithms when there are many stragglers. A typical result is shown in Figure 5.1. In it, we observe that SGC and ErasureHead outperform Ignore–Stragglers–SGD at the expense of doubling the redundancy. In Figure 5.2 we plot the convergence of approximate gradient codes as function of p. We observe that SGC outperforms ErasureHead when the number of stragglers is large, $p > 0.6$. As expected, the approximate algorithms have worse accuracy than full-blown gradient descent, but we note that implementing exact gradient descent with a p fraction of stragglers would require redundancy $d \approx pn \gg 2$. Moreover, we observe the flexibility of the approximate algorithms in the number of stragglers, and we note that computing GD exactly would lack this flexibility. In Section 5.6, we comment on how the dependency between stragglers affect the convergence of SGC. Our implementation is publicly available [147].

5.5 Theoretical Results

In this section we precisely state our theoretical results. We begin with a specialized result for the ℓ_2 loss function, and then include a result for more general loss functions.

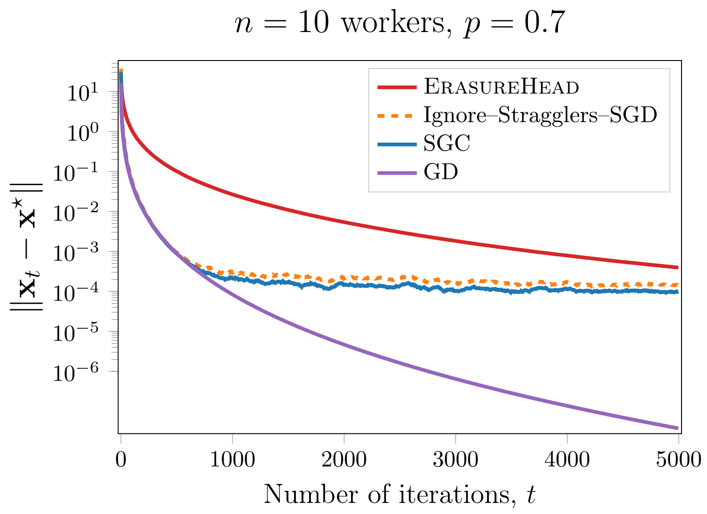


Figure 5.1: Comparison between Ignore–Stragglers–SGD, SGC and ErasureHead in terms of the distance between $\mathbf{x}_t$ and $\mathbf{x}^\star$ the value of $\mathbf{x}$ that minimizes the loss function. SGC outperforms Ignore–Straggler–SGD at the expense of adding small redundancy, $d = 2$ in this example.

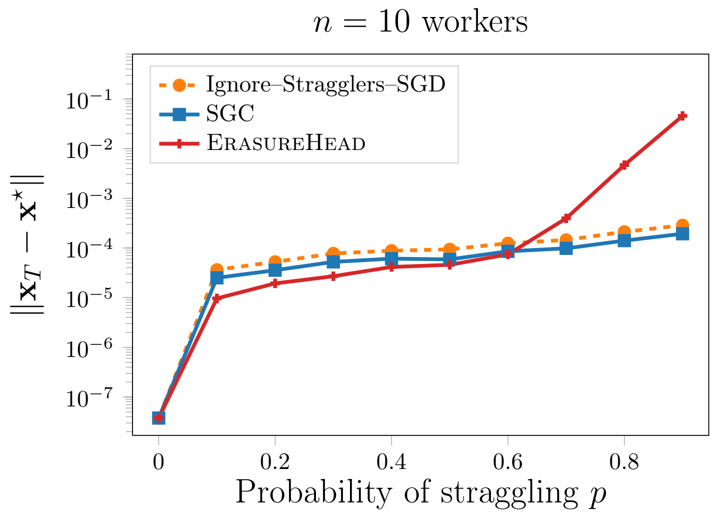


Figure 5.2: Final convergence of all algorithms run for $T = 5000$ iterations as function of p the probability of workers being stragglers. We omit GD in this setting, because it has the same performance as all algorithms when $p = 0$.

5.5.1 Special case: ℓ_2 loss function

We begin with a result that holds for the special case of an ℓ_2 loss function, aka, regression. Inspired by the approach of [146] for SGD, our approach is to consider a *weighted* distribution scheme; that is, we choose d_i proportionally to $\|\mathbf{b}_i\|_2^2$. While the statement below is only for the ℓ_2 loss function, we conjecture that it holds for more general loss functions.

Define a parameter

$$\mu = \frac{\frac{1}{m}\|B\|_F^2}{\|B^TB\|}.$$

This parameter measures how incoherent B is. If B is orthogonal, $\mu = 1$, while if, for example, B is the all-ones matrix, then $\mu = 1/m$. It is not hard to check that $\mu \in [0,1]$.

Suppose that $\mathcal{D}$ is a pair-wise balanced distribution scheme which sends $\mathbf{a}_i$ to d_i different workers, where

$$d_i = \sigma \cdot \|\mathbf{b}_i\|_2^2, \tag{5.6}$$

$$\sigma = \frac{md}{\|B\|_F^2} = \frac{d}{\mu\|B^TB\|}, \tag{5.7}$$

$$d = \frac{1}{m}\sum_{i\in[m]} d_i. \tag{5.8}$$

The parameter d is the average redundancy of the scheme that will control σ and the d_i's. Notice that, as stated, it is possible that the d_i end up being non-integers; in the following, we will assume for simplicity below that $d_i \in \mathbb{Z}$ for all i. Notice that if $\|\mathbf{b}_i\|_2 = 1$ for all i, then this will be the case because we can choose $d_i = d$ to be any integer of our choice, and this defines σ.

Theorem 5.6. *Consider an SGC algorithm run on a matrix $A \triangleq [B|\mathbf{y}]$ of dimension $m\times(\ell+1)$ distributed to n workers according to a pairwise balanced distribution scheme with d_i as described above, with loss function*

$$\mathcal{L}([B|\mathbf{y}], \mathbf{x}) = \|B\mathbf{x} - \mathbf{y}\|_2^2,$$

and assume that the degrees $d_i \leq n$ are all integers.

Suppose the stragglers follow the stochastic model of Section 5.2, and that each worker is a straggler independently with probability p. Choose $\varepsilon > 0$ and choose $T \geq 2\log(1/\varepsilon^2)$.

Suppose that the number of workers n satisfies $n \geq 8\left(\frac{p}{1-p}\right)$, and that

$$8\mu\left(\frac{p}{1-p}\right) \leq d.$$

Choose a step size

$$\gamma_t = \frac{1}{\|B^TB\|} \cdot \min\left\{\frac{1}{2}, \frac{\log(1/\varepsilon^2)}{t}\right\}.$$

Then, after T iterations of SGC, we have

$$\mathbb{E}\left[\|\mathbf{x}_T - \mathbf{x}^\star\|_2^2\right] \leq \varepsilon^2\|\mathbf{x}_0 - \mathbf{x}^\star\|_2^2 + \frac{1}{dT}\left(\log^2(1/\varepsilon^2)\left(\frac{p}{1-p}\right)\|\tilde{\mathbf{r}}\|^2\mu\right)$$

where the expectation is over the stragglers in each of the T iterations of SGC and where μ is as above, and where

$$\tilde{\mathbf{r}} = \frac{\|B\mathbf{x}^\star - \mathbf{y}\|_2^2}{\|B^TB\|_2}.$$

Corollary 5.7. *Suppose that $B\mathbf{x}^\star = \mathbf{y}$ (that is, we are solving a system for which there is a solution) and that $n \geq 8p/(1-p)$. Then the algorithm described in Theorem 5.6 converges with*

$$\mathbb{E}\left[\|\mathbf{x}_T - \mathbf{x}^\star\|_2^2\right] \leq \varepsilon^2\|\mathbf{x}_0 - \mathbf{x}^\star\|^2$$

provided that $T \geq 2\log(1/\varepsilon^2)$ and $d \geq 8\mu p/(1-p)$.

In particular, since $\mu \leq 1$, this says that we need to take $d \gtrsim p/(1-p)$ and the algorithm converges extremely quickly.

5.5.2 Beyond ℓ_2 loss function

Now, we consider a constrained version of the problem given in (5.1), where $\mathbf{x}$ belongs to a bounded set $\mathcal{W}$. In this section, we state a result for general loss functions $\mathcal{L}$ which are λ-strongly convex:

Definition 5.8 (Strongly convex function)**.** A function $\mathcal{L}$ is λ-strongly convex, if for all $\mathbf{x}$, $\mathbf{x}'$ $\in \mathbb{R}^\ell$ and any subgradient $\mathbf{g}$ of $\mathcal{L}$ at $\mathbf{x}$,

$$\mathcal{L}(\mathbf{x}') \geq \mathcal{L}(\mathbf{x}) + \langle \mathbf{g}, \mathbf{x}' - \mathbf{x} \rangle + \frac{\lambda}{2} \left\| \mathbf{x}' - \mathbf{x} \right\|_2^2. \tag{5.9}$$

Theorem 5.9 below follows from the analysis in [36].

Theorem 5.9. *Suppose that SGC is run on a matrix $A \triangleq [B|\mathbf{y}]$ of dimension $m \times (\ell + 1)$ distributed to n workers with each row of A sent to d_i different workers, $d_i \leq n$, according to a pairwise balanced distribution scheme. Consider a version of the optimization problem in (5.1) where $\mathbf{x}$ is constrained to a convex set $\mathcal{W}$, i.e.,*

$$\mathbf{x}^\star = \arg \min_{\mathbf{x} \in \mathcal{W}} \mathcal{L}(A, \mathbf{x}),$$

and at each step of the algorithm $\mathbf{x}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{x}_t - \gamma_t \hat{\mathbf{g}}_t)$, where Π is the projection operator. Let p denote the probability of a given worker being a straggler at a given iteration. Suppose that the loss function $\mathcal{L}$ is λ-strongly convex with respect to the optimal point $\mathbf{x}^\star \in \mathcal{W}$, and that all of the partial gradients $\nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x})$ are bounded for $i \in [n]$ and $\mathbf{x} \in \mathcal{W}$, i.e. there exists a constant C so that

$$\|\nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x})\|_2^2 \leq C, \qquad \forall \mathbf{x} \in \mathcal{W}, i \in [n].$$

Suppose that the step size is set to be $\gamma_t = 1/(\lambda t)$. Then the error after T iterations is bounded by

$$\mathbb{E}\|\mathbf{x}_T - \mathbf{x}^\star\|_2^2 \leq \frac{4}{\lambda^2 T} \cdot mC^2 \left(\frac{p}{1-p} \cdot \frac{1}{d_{\min}} + \frac{(m-1)p}{n(1-p)} + m \right) \tag{5.10}$$

Algorithm	Brief description
Stochastic Gradient Code (SGC)	The master sends each data vector $\mathbf{b}_i$ to d_i workers chosen at random, where d_i is proportional to the ℓ_2 norm of $\mathbf{b}_i$ and is computed as in (5.6) with $d = 2$. Each worker sends to the master the weighted sum of its partial gradients as in (5.4). The master computes the gradient estimate as the sum of the received results from non straggling workers.
Bernoulli Gradient Code (BGC) [14]	Similar to SGC but all data vectors are replicated d times, i.e., $d_i = 2$ for all $i \in [m]$.
ErasureHead [14, 17]	Partitions the data set equally and sends each partition to d workers. Workers send the sum of the partial gradients to the master who computes the gradient estimate as the sum of distinct received partial gradients divided by total number of data vectors.
Ignore–Stragglers–SGD	Partitions the data among the workers with no redundancy. Workers send the sum of the partial gradients to the master who computes the gradient estimate as the sum of distinct partial gradients divided by the average number of data vectors received per iteration.
SGC–Send–All	Same as SGC with one difference: at each iteration the workers send all the partial gradients to the master. The master computes the gradient estimate as the sum of *distinct* partial gradients divided by the average number of data vectors received per iteration.

Table 5.3: Summary of the stochastic algorithms that we implement in our simulations.

where $d_{min} \triangleq \min_{i\in[m]} d_i$.

This shows that SGC does have a convergence rate of $O(1/T)$, matching regular SGD [36, Lemma 1]. This means that at least the convergence rate is not hurt by the fact that the data assignment is fixed. However, unlike Theorem 5.6, this result does not always significantly improve as d increases (although we note that the bound above *is* decreasing in $d_{\min}$, so in some parameter regimes—when $n \gg m$ and p is close to 1—this does indicate some improvement). We leave it as an interesting open problem to fully generalize our result of Theorem 5.6 to general loss functions.

5.6 Simulation Results

5.6.1 Simulation setup

We simulated the performance of SGC on synthetic data B of dimension 1000×100. The data is generated as follows: each row vector $\mathbf{b}_i$ is generated using a Gaussian distribution $\mathcal{N}(0, 100)$. We pick a random vector $\bar{\mathbf{x}}$ with components being integers between 1 and 10 and generate $y_i \sim \mathcal{N}(\langle \mathbf{b}_i, \bar{\mathbf{x}} \rangle, 1)$. Our code and the generated data set can be found in [147].

We run linear regression using the ℓ_2 loss function, i.e.,

$$\mathcal{L}(\mathbf{a}_i, \mathbf{x}_t) = \frac{1}{2}\left(\langle \mathbf{b}_i, \mathbf{x}_t\rangle - y_i\right)^2.$$

We show simulations for $n = 10$ workers. For each simulation we vary the probability of a worker being a straggler from $p = 0$ to $p = 0.9$ with a step of 0.1. We run the algorithm for 5000 iterations with a variable step size given[6] by

$$\gamma_t = 7\frac{\ln(10^{100})}{t^{0.7}}. \tag{5.11}$$

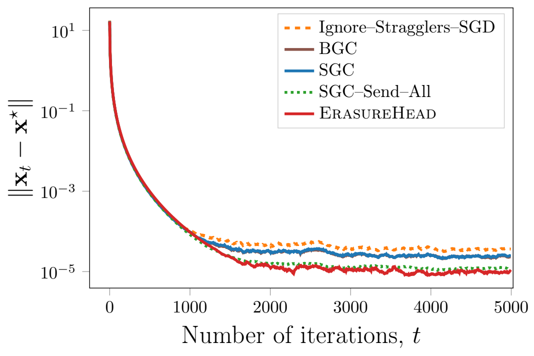
For all simulations, we run each experiment 10 times and average the results. For SGC, each data vector $\mathbf{a}_i$ is replicated d_i times, where the d_i's are computed as in (5.6) and (5.7) with $d = 2$. Then, each d_i is rounded to the nearest integer. Due to rounding, the actual value of d given in (5.8) will be close to 2. In our generated data set, the majority of the d_i's are equal to 2 while the others are either 1 or 3 resulting in average redundancy $d = 2.024$. For the other algorithms in Table 5.3, the average redundancy d is chosen to be exactly equal to 2.

We omit comparing SGC to the gradient codes in [13] and [40] because they do not match our setting; the former requires a high redundancy factor d and the latters requires the master to run a decoding algorithm at each iteration.

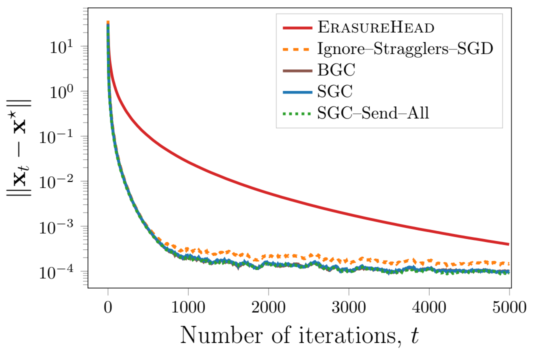
5.6.2 Convergence

In Figures 5.3a and 5.3b, we plot the error $\|\mathbf{x}_t - \mathbf{x}^\star\|$ for up to 5000 iterations for small and large probability of workers being stragglers, namely for $p = 0.1$ and $p = 0.7$, respectively. Here, $\mathbf{x}^\star$ given in (5.1) is computed using the pseudoinverse of B, i.e., $\mathbf{x}^\star = \left(B^T B\right)^{-1} B^T Y$. We notice that for all p the convergence rate of SGC exhibits two phases: an exponential decay followed by an error floor. To see the benefit of replication, we compare SGC to Ignore–Stragglers–SGD.

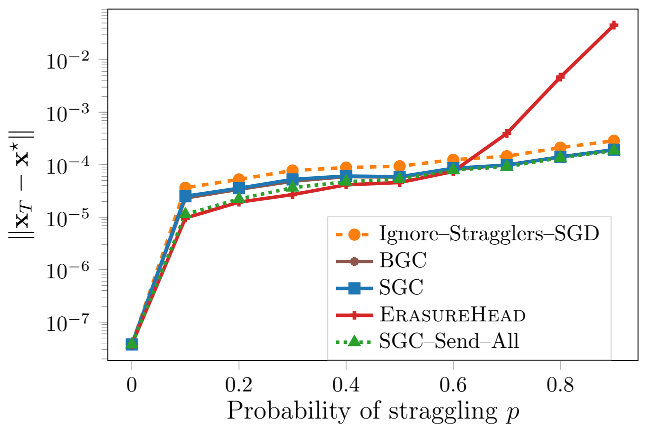
[6]In our theoretical analysis we assumed that the step size γ_t is proportional to $1/t$. In our numerical simulations, we tried different functions of γ_t and observed that the one in (5.11) gives better convergence rate for all the considered algorithms.



(a) Error vs iterations, $p = 0.1$.



(b) Error vs iterations, $p = 0.7$.



(c) Error at $T = 5000$ vs p.

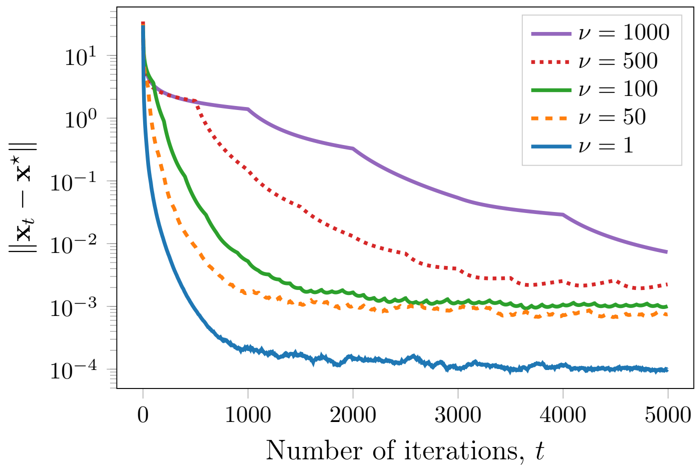
Figure 5.3: Convergence as function of probability of workers being stragglers p is shown for small $p = 0.1$ in (a) and for large $p = 0.7$ in (b) for $n = 10$ workers. SGC convergence has two phases: an exponential decay in the beginning until it reaches an error floor. SGC has same performance as BGC, but outperforms ErasureHead for large values of p. In (c) the error floor at $T = 5000$ iterations is shown versus p.

Both have the same performance in the exponential phase, but SGC has a lower error floor due to redundancy. A lower bound on the performance of SGC is SGC–Send–All which has a lower error floor because it computes a better estimate of the gradient at the expense of a higher communication cost. However, as p increases the gap between the two error floor of SGC and SGC–Send–All decreases. In our simulations, we notice the error floor of both algorithms almost match for $p \geq 0.6$ as can be seen in Figure 5.3c.

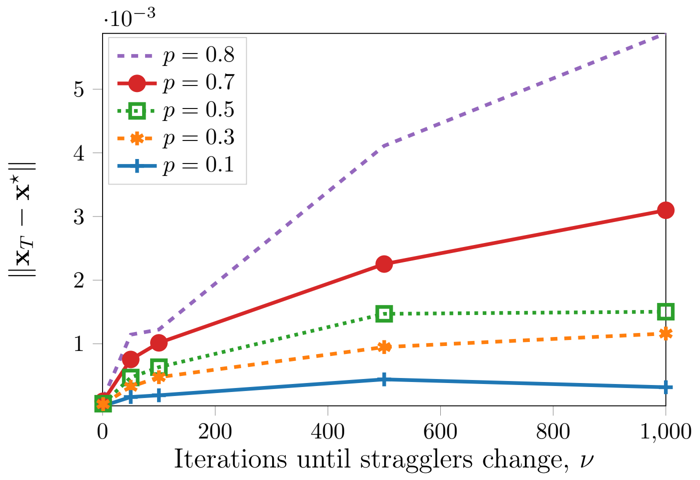
For our chosen data set, SGC and BGC have similar performance. This is mainly due to the fact that most of the data vectors have the same replication factor in BGC and SGC which is 2 times. For other data sets with more variance in the d_i's, we observe that SGC can have better performance. ErasureHead has better error floor than SGC for small values of p. However, for large p the rate of the exponential decay drastically decreases for ErasureHead.

5.6.3 Dependency between stragglers across iterations

Our theoretical analysis assumes that the stragglers are independent across iterations. We check the effect of this dependency on the numerical performance of SGC. We use a simple model to enforce dependency of stragglers across iterations by fixing the stragglers for ν iterations, after which the stragglers are chosen again randomly and iid with a probability p and this is repeated until the algorithm stops. The special value of $\nu = 1$ implies that the stragglers are independent. A large value of ν implies a longer dependency among the stragglers across iterations. We observe in Figure 5.4 that SGC still maintains the two phases behavior. However, as ν increases, the rate of convergence decreases and the error floor increases.



(a) Error versus iterations, $p = 0.7$.



(b) Error at iteration $T = 5000$ versus ν.

Figure 5.4: The effect of the dependency of stragglers across iterations on the performance of SGC. We assume that the identity of the stragglers change every ν iterations. In (a) the convergence of the error as function of the number of iterations is shown for different values of ν and $p = 0.7$. SGC maintains an exponential decay in the error for the tested values of ν and p. However, the rate of the decay decreases and the error floor increases with the increase of ν. In (b), the error at iteration $T = 5000$ as function of ν is shown for different values of p.

5.7 Proofs

5.7.1 Proof of Theorem 5.6

In this section, we prove Theorem 5.6. In the case when the loss function is

$$\mathcal{L}([B|y], \mathbf{x}) = \frac{1}{2}\|B\mathbf{x} - \mathbf{y}\|_2^2,$$

we have

$$\nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x}) = (\langle \mathbf{b}_i, \mathbf{x} \rangle - y_i) \cdot \mathbf{b}_i,$$

so that

$$\sum_i^m \nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x}) = B^T(B\mathbf{x} - \mathbf{y}) = \nabla \mathcal{L}(A, \mathbf{x}).$$

Fix an iteration t. Let Z_i (which depends on t; we suppress this dependence in the notation) be defined by

$$Z_i = \sum_{j=1}^n \mathcal{I}_i^j.$$

That is, Z_i is the number of workers who hold $\mathbf{a}_i$ who are not stragglers at round t. Thus, Z_i is a binomial random variable with mean $d_i(1-p)$ and variance $d_i p(1-p)$. Let

$$\tilde{Z}_i = Z_i - \mathbb{E}\left[Z_i\right],$$

so that

$$\mathbb{E}\left[\tilde{Z}_i^2\right] = d_i p(1-p)$$

and

$$\mathbb{E}\left[\tilde{Z}_i \tilde{Z}_j\right] = \frac{d_i d_j}{n} p(1-p).$$

From the definition of $\mathbf{x}_{t+1}$ and replacing $\hat{\mathbf{g}}_t$ by its value from (5.5), we have

$$\begin{aligned}
\mathbf{x}_{t+1} &= \mathbf{x}_t - \gamma_t \hat{\mathbf{g}}_t \\
&= \mathbf{x}_t - \gamma_t \sum_{j=1}^{n} \sum_{i=1}^{m} \frac{\mathcal{I}_i^j}{d_i(1-p)} \nabla \mathcal{L}(\mathbf{a}_i, \mathbf{x}) \\
&= \mathbf{x}_t - \gamma_t \sum_{i=1}^{m} \frac{Z_i}{d_i(1-p)} (\langle \mathbf{b}_i, \mathbf{x}_t \rangle - y_i) \mathbf{b}_i \\
&= \mathbf{x}_t - \sum_{i=1}^{m} Z_i \delta_i (\langle \mathbf{b}_i, \mathbf{x}_t \rangle - y_i) \mathbf{b}_i,
\end{aligned}$$

where we define

$$\delta_i = \frac{\gamma_t}{d_i(1-p)}.$$

(Notice that δ_i also depends on t; we suppress this for notational convenience).

Expanding out the terms, we have

$$\begin{aligned}
\mathbf{x}_{t+1} - \mathbf{x}^\star = \mathbf{x}_t - \mathbf{x}^\star &- \sum_{i=1}^{m} \mathbb{E}\left[Z_i\right] \delta_i \mathbf{b}_i \mathbf{b}_i^T (\mathbf{x}_t - \mathbf{x}^\star) \\
&- \sum_{i=1}^{m} \tilde{Z}_i \delta_i \mathbf{b}_i \mathbf{b}_i^T (\mathbf{x}_t - \mathbf{x}^\star) \\
&- \sum_{i=1}^{m} \mathbb{E}\left[Z_i\right] \delta_i (\langle \mathbf{b}_i, \mathbf{x}^\star \rangle - y_i) \mathbf{b}_i \\
&- \sum_{i=1}^{m} \tilde{Z}_i \delta_i (\langle \mathbf{b}_i, \mathbf{x}^\star \rangle - y_i) \mathbf{b}_i
\end{aligned}$$

where we have split up $Z_i = \mathbb{E}\left[Z_i\right] + \tilde{Z}_i$ and

$$(\langle \mathbf{b}_i, \mathbf{x}_t \rangle - y_i) \mathbf{b}_i = \mathbf{b}_i \mathbf{b}_i^T (\mathbf{x}_t - \mathbf{x}^\star) + (\langle \mathbf{b}_i, \mathbf{x}^\star \rangle - y_i) \mathbf{b}_i.$$

Letting

$$\mathbf{r} := B\mathbf{x}^\star - \mathbf{y}$$

be the optimal residual and writing the above in matrix notation, we have

$$\mathbf{x}_{t+1}-\mathbf{x}^\star = (\mathbf{x}_t-\mathbf{x}^\star)-(1-p)B^TD_dD_\delta B(\mathbf{x}_t-\mathbf{x}^\star)-B^TD_{\tilde{Z}}D_\delta B(\mathbf{x}_t-\mathbf{x}^\star)-(1-p)B^TD_dD_\delta\mathbf{r}-B^TD_{\tilde{Z}}D_\delta\mathbf{r},$$

where $D_{\tilde{Z}}$ is diagonal with entries $\tilde{Z}_i$, D_δ is diagonal with entries δ_i, and D_d is diagonal with entries d_i. Recalling that

$$\delta_i = \frac{\gamma_t}{(1-p)d_i}$$

we have

$$D_\delta \cdot D_d = \frac{\gamma_t}{1-p}.$$

Thus we can simplify the above as

$$\begin{aligned}\mathbf{x}_{t+1}-\mathbf{x}^\star &= (\mathbf{x}_t-\mathbf{x}^\star)-\gamma_tB^TB(\mathbf{x}_t-\mathbf{x}^\star)-B^TD_{\tilde{Z}}D_\delta B(\mathbf{x}_t-\mathbf{x}^\star)-\gamma_tB^T\mathbf{r}-B^TD_{\tilde{Z}}D_\delta\mathbf{r}\\ &= (\mathbf{x}_t-\mathbf{x}^\star)-\gamma_tB^TB(\mathbf{x}_t-\mathbf{x}^\star)-B^TD_{\tilde{Z}}D_\delta B(\mathbf{x}_t-\mathbf{x}^\star)-B^TD_{\tilde{Z}}D_\delta\mathbf{r}\end{aligned}$$

using the fact that $B^T\mathbf{r}=0$ since $\mathbf{r}=B\mathbf{x}^\star-\mathbf{y}$ is the optimal residual. We simplify this further as:

$$\mathbf{x}_{t+1}-\mathbf{x}^\star = (I-\gamma_tB^TB+B^TD_{\tilde{Z}}D_\delta B)(\mathbf{x}_t-\mathbf{x}^\star)-B^TD_{\tilde{Z}}D_\delta\mathbf{r}.$$

Now we compute $\mathbb{E}\left[\|\mathbf{x}_{t+1}-\mathbf{x}^\star\|^2\right]$, where the expectation is over the choice of $\mathbf{x}_{t+1}$, conditioned on $\mathbf{x}_t$. We have

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{x}_{t+1}-\mathbf{x}^\star\|^2\right] &= (\mathbf{x}_t-\mathbf{x}^\star)^T\left[(I-\gamma_tB^TB)^2+B^TD_\delta\mathbb{E}[D_{\tilde{Z}}BB^TD_{\tilde{Z}}]D_\delta B\right](\mathbf{x}_t-\mathbf{x}^\star) &(5.12)\\ &+\mathbf{r}^TD_\delta\mathbb{E}[D_{\tilde{Z}}BB^TD_{\tilde{Z}}]D_\delta\mathbf{r} &(5.13)\\ &+\mathbf{r}^TD_\delta\mathbb{E}[D_{\tilde{Z}}BB^TD_{\tilde{Z}}]D_\delta B(\mathbf{x}_t-\mathbf{x}^\star) &(5.14)\\ &+\mathbf{r}^TD_\delta\mathbb{E}[D_{\tilde{Z}}]BB^T(I-\gamma_tB^TB)(\mathbf{x}_t-\mathbf{x}^\star) &(5.15)\\ &+(\mathbf{x}_t-\mathbf{x}^\star)^T(I-\gamma_tB^TB)(B^T\mathbb{E}[D_{\tilde{Z}}]D_\delta B)(\mathbf{x}_t-\mathbf{x}^\star). &(5.16)\end{aligned}$$

We handle each of these terms below. First, we observe that (5.15) and (5.16) are zero because $\mathbb{E}D_{\tilde{Z}} = 0$. In order to handle (5.12), (5.13), (5.14), we compute

$$\mathbb{E}\left[D_{\tilde{Z}}BB^TD_{\tilde{Z}}\right].$$

The off-diagonal elements are given by

$$\mathbb{E}\left[\tilde{Z}_i\tilde{Z}_j\left\langle \mathbf{b}_i,\mathbf{b}_j\right\rangle\right] = \frac{d_id_j}{n}p(1-p)\left\langle \mathbf{b}_i,\mathbf{b}_j\right\rangle,$$

and the diagonal elements are given by

$$\mathbb{E}\left[\tilde{Z}_i^2\|\mathbf{b}_i\|^2\right] = d_ip(1-p)\|\mathbf{b}_i\|^2 = d_i^2p(1-p)/\sigma.$$

Thus,

$$\mathbb{E}\left[D_{\tilde{Z}}BB^TD_{\tilde{Z}}\right] = p(1-p)\left(\frac{1}{n}D_dBB^TD_d + \frac{1}{\sigma}\left(I-\frac{D_d}{n}\right)D_d^2\right).$$

Now we handle the terms (5.12) and (5.13). First, for (5.12), we have

$$\begin{aligned}
&(\mathbf{x}_t-\mathbf{x}^\star)^T\left[(I-\gamma_tB^TB)^2 + B^TD_\delta\mathbb{E}\left[D_{\tilde{Z}}BB^TD_{\tilde{Z}}\right]D_\delta B\right](\mathbf{x}_t-\mathbf{x}^\star)\\
&=(\mathbf{x}_t-\mathbf{x}^\star)^T\left[(I-\gamma_tB^TB)^2 + p(1-p)B^TD_\delta\left(\frac{1}{n}D_dBB^TD_d+\frac{1}{\sigma}\left(I-\frac{D_d}{n}\right)D_d^2\right)D_\delta B\right](\mathbf{x}_t-\mathbf{x}^\star)\\
&=(\mathbf{x}_t-\mathbf{x}^\star)^T\left[(I-\gamma_tB^TB)^2 + \frac{p(1-p)}{n}B^TD_\delta D_dBB^TD_dD_\delta B + \frac{p(1-p)}{\sigma}B^T\left(I-\frac{D_d}{n}\right)D_\delta D_d^2D_\delta B\right](\mathbf{x}_t-\mathbf{x}^\star)\\
&=(\mathbf{x}_t-\mathbf{x}^\star)^T\left[(I-\gamma_tB^TB)^2 + \frac{p(1-p)\gamma_t^2}{n}B^TBB^TB + \frac{\gamma_t^2p}{(1-p)\sigma}B^T\left(I-\frac{D_d}{n}\right)B\right](\mathbf{x}_t-\mathbf{x}^\star),
\end{aligned}$$

where in the last line we used the fact again that $D_\delta D_d = \gamma_tI/(1-p)$. Now we can bound this term by

$$(5.12) \leq \left((1-\gamma_t\|B^TB\|)^2 + \frac{p\gamma_t^2}{(1-p)n}\|B^TB\|^2 + \frac{\gamma_t^2p}{(1-p)\sigma}\|B^TB\|\right)\|\mathbf{x}_t-\mathbf{x}^\star\|^2,$$

where above we have used the fact that

$$\left\| B^T \left(I - \frac{D_d}{n} \right) B \right\| \leq \|B^T B\|,$$

because $I - D_d/n$ is a diagonal matrix whose diagonal entries are all in $[0,1]$ (using the fact that $d_i \leq n$ for all i). The second term (5.13) is bounded by

$$\begin{aligned}
(5.13) &= \mathbf{r}^T D_\delta \mathbb{E}\left[D_{\tilde{Z}} B B^T D_{\tilde{Z}}\right] D_\delta \mathbf{r} \\
&= p(1-p)\mathbf{r}^T D_\delta \left(\frac{1}{n} D_d B B^T D_d + \left(I - \frac{D_d}{n} \right) \frac{1}{\sigma} D_d^2 \right) D_\delta \mathbf{r} \\
&= \frac{p(1-p)}{n} \mathbf{r}^T D_\delta D_d B B^T D_d D_\delta \mathbf{r} + \frac{p(1-p)}{\sigma} \mathbf{r}^T \left(I - \frac{D_d}{n} \right) D_\delta D_d^2 D_\delta \mathbf{r} \\
&\leq \frac{\gamma_t^2 p}{(1-p)n} \mathbf{r}^T B B^T \mathbf{r} + \gamma_t^2 \cdot \frac{p}{1-p} \cdot \frac{\mathbf{r}^T \left(I - \frac{D_d}{n} \right) \mathbf{r}}{\sigma} \\
&\leq \gamma_t^2 \cdot \frac{p}{1-p} \cdot \frac{\|\mathbf{r}\|^2}{\sigma},
\end{aligned}$$

where we have used the fact that $B^T \mathbf{r} = 0$, and that $\mathbf{r}^T \left(I - D_d/n \right) \mathbf{r} \leq \|\mathbf{r}\|^2$ because $d_i \leq n$ for all i.

Finally we bound (5.14). We have, using our expression for $\mathbb{E}[D_{\tilde{Z}} B B^T D_{\tilde{Z}}]$ from above that

$$\begin{aligned}
(5.14) &= \mathbf{r}^T D_\delta \mathbb{E}[D_{\tilde{Z}} B B^T D_{\tilde{Z}}] D_\delta B(\mathbf{x}_t - \mathbf{x}^\star) \\
&= p(1-p)\mathbf{r}^T D_\delta \left(\frac{1}{n} D_d B B^T D_d + \frac{1}{\sigma} \left(I - \frac{D_d}{n} \right) D_d^2 \right) D_\delta B(\mathbf{x}_t - \mathbf{x}^\star) \\
&= \frac{p(1-p)}{n} \mathbf{r}^T D_\delta D_d B B^T D_d D_\delta B(\mathbf{x}_t - \mathbf{x}^\star) + \frac{p(1-p)}{\sigma} \mathbf{r}^T D_\delta \left(I - \frac{D_d}{n} \right) D_d^2 D_\delta B(\mathbf{x}_t - \mathbf{x}^\star) \\
&= \frac{\gamma_t^2 p}{(1-p)n} \mathbf{r}^T B B^T B(\mathbf{x}_t - \mathbf{x}^\star) + \frac{\gamma_t^2 p}{(1-p)\sigma} \mathbf{r}^T \left(I - \frac{D_d}{n} \right) B(\mathbf{x}_t - \mathbf{x}^\star)
\end{aligned}$$

using the fact that $D_\delta D_d = \gamma_t I/(1-p)$ in the last line. Now, the first term is equal to zero because $\mathbf{r}^T B = 0$, and we have

$$\begin{aligned}(5.14) &= \frac{\gamma_t^2 p}{(1-p)\sigma}\mathbf{r}^T\left(I - \frac{D_d}{n}\right)B(\mathbf{x}_t - \mathbf{x}^\star)\\ &= \frac{\gamma_t^2 p}{(1-p)\sigma n}\mathbf{r}^T D_d B(\mathbf{x}_t - \mathbf{x}^\star)\end{aligned}$$

again using the fact that $\mathbf{r}^T B = 0$. Finally, we can bound

$$\begin{aligned}(5.14) &= \frac{\gamma_t^2 p}{(1-p)\sigma n}\mathbf{r}^T D_d B(\mathbf{x}_t - \mathbf{x}^\star)\\ &\le \frac{\gamma_t^2 p}{(1-p)\sigma n}\|\mathbf{r}\|\|D_d B(\mathbf{x}_t - \mathbf{x}^\star)\|\\ &\le \frac{\gamma_t^2 p}{(1-p)\sigma}\|\mathbf{r}\|\|B(\mathbf{x}_t - \mathbf{x}^\star)\|\\ &\le \frac{\gamma_t^2 p}{(1-p)\sigma}\|\mathbf{r}\|\sqrt{\|B^T B\|}\|\mathbf{x}_t - \mathbf{x}^\star\|\\ &\le \frac{\gamma_t^2 p}{(1-p)\sigma}\left(\frac{\|\mathbf{r}\|^2 + \|B^T B\|\|\mathbf{x}_t - \mathbf{x}^\star\|^2}{2}\right),\end{aligned}$$

using the arithmetic-geometric-mean inequality in the final line. In particular, this term is similar to terms that appear in both (5.12) and (5.13), and (along with the observation that (5.15), (5.16) are zero) we have

$$\begin{aligned}\mathbb{E}\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 &\le (5.12) + (5.13) + (5.14)\\ &\le \left((1-\gamma_t\|B^T B\|)^2 + \frac{p\gamma_t^2}{(1-p)n}\|B^T B\|^2 + \frac{\gamma_t^2 p}{(1-p)\sigma}\|B^T B\|\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2\\ &\quad + \gamma_t^2 \cdot \frac{p}{1-p}\cdot\frac{\|\mathbf{r}\|^2}{\sigma}\\ &\quad + \frac{\gamma_t^2 p}{(1-p)\sigma}\left(\frac{\|\mathbf{r}\|^2 + \|B^T B\|\|\mathbf{x}_t - \mathbf{x}^\star\|^2}{2}\right)\\ &\le \left((1-\gamma_t\|B^T B\|)^2 + \frac{p\gamma_t^2}{(1-p)n}\|B^T B\|^2 + \frac{2\gamma_t^2 p}{(1-p)\sigma}\|B^T B\|\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \quad (5.17)\\ &\quad + 2\gamma_t^2 \cdot \frac{p}{1-p}\cdot\frac{\|\mathbf{r}\|^2}{\sigma}. \quad (5.18)\end{aligned}$$

Now we recall our choice of

$$\gamma_t = \frac{1}{\|B^T B\|} \cdot \min\left\{\frac{1}{2}, \frac{\log(1/\varepsilon^2)}{t}\right\},$$

and the definition of

$$\sigma = \frac{d}{\mu}\frac{1}{\|B^T B\|}.$$

Let

$$\tilde{\gamma}_t := \|B^T B\|\gamma_t = \min\left\{\frac{1}{2}, \frac{\log(1/\varepsilon^2}{t}\right\}.$$

Now we can simplify our bounds on (5.17) and (5.18) as:

$$\begin{aligned}
(5.17) &\le \left((1-\gamma_t\|B^T B\|)^2 + \frac{p\gamma_t^2}{(1-p)n}\|B^T B\|^2 + 2\frac{\gamma_t^2 p}{(1-p)\sigma}\|B^T B\|\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \\
&\le \left((1-\tilde{\gamma}_t)^2 + \left(\frac{p}{1-p}\right)(\tilde{\gamma}_t)^2 \cdot \frac{1}{n} + \left(\frac{2p}{1-p}\right)(\tilde{\gamma}_t)^2\left(\frac{\mu}{d}\right)\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \\
&\le \left((1-\tilde{\gamma}_t)^2 + \frac{1}{2}(\tilde{\gamma}_t)^2\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2,
\end{aligned}$$

using the assumptions that $n \ge 4p/(1-p)$ and $d \ge 8\mu p/(1-p)$. Now we have:

$$\begin{aligned}
(5.17) &\le \left((1-\tilde{\gamma}_t)^2 + \frac{1}{2}(\tilde{\gamma}_t)^2\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \\
&= \left(1 - 2\tilde{\gamma}_t + \frac{3}{2}\tilde{\gamma}_t\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \\
&\le (1-\tilde{\gamma}_t)\|\mathbf{x}_t - \mathbf{x}^\star\|^2,
\end{aligned}$$

using from the definition of $\tilde{\gamma}_t$ that $\tilde{\gamma}_t \le 1/2$ and hence $\tilde{\gamma}_t^2 \le \frac{1}{2}\tilde{\gamma}_t$.

Meanwhile,

$$\begin{aligned}(5.18) &\leq 2\gamma_t^2 \cdot \frac{p}{1-p}\frac{\|\mathbf{r}\|^2}{\sigma} \\ &\leq 2\left(\tilde{\gamma}_t\right)^2\left(\frac{p}{1-p}\right)\frac{\|\mathbf{r}\|^2}{\sigma\|B^TB\|^2} \\ &\leq 2\left(\tilde{\gamma}_t\right)^2\left(\frac{p}{1-p}\right)\frac{\|\tilde{\mathbf{r}}\|^2}{\sigma\|B^TB\|},\end{aligned}$$

recalling that $\tilde{\mathbf{r}} = \mathbf{r}/\|B^TB\|$. Thus

$$\begin{aligned}(5.18) &\leq 2\left(\tilde{\gamma}_t\right)^2\left(\frac{p}{1-p}\right)\frac{\|\tilde{\mathbf{r}}\|^2}{\sigma\|B^TB\|} \\ &= 2\left(\tilde{\gamma}_t\right)^2\left(\frac{p}{1-p}\right)\left(\frac{\mu}{d}\right)\|\tilde{\mathbf{r}}\|^2.\end{aligned}$$

Putting the two terms together, we conclude that for fixed t,

$$\mathbb{E}\left[\|\mathbf{x}_{t+1}-\mathbf{x}^\star\|_2^2\right] \leq \left(1-\tilde{\gamma}_t\right)\|\mathbf{x}_t-\mathbf{x}^\star\|_2^2 + 2\left(\tilde{\gamma}_t\right)^2\left(\frac{p}{1-p}\right)\left(\frac{\mu}{d}\right)\|\tilde{\mathbf{r}}\|^2.$$

Now, we proceed by induction, using the fact that the stragglers are independent between the different rounds, to conclude that

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{x}_T-\mathbf{x}^\star\|_2^2\right] &\leq \left(\prod_{t=1}^{T}(1-\tilde{\gamma}_t)\right)\|\mathbf{x}_0-\mathbf{x}^\star\|_2^2 + 2T\tilde{\gamma}_T^2\left(\frac{p}{1-p}\right)\left(\frac{\mu}{d}\right)\|\tilde{\mathbf{r}}\|_2^2 \\ &\leq \left(1-\frac{\log(1/\varepsilon^2)}{T}\right)^T\|\mathbf{x}_0-\mathbf{x}^\star\|_2^2 + 2T\cdot\frac{\log^2(1/\varepsilon^2)}{T^2}\left(\frac{p}{1-p}\right)\left(\frac{\mu}{d}\right)\|\tilde{\mathbf{r}}\|_2^2 \\ &\leq \varepsilon^2\|\mathbf{x}_0-\mathbf{x}^\star\|_2^2 + \frac{2}{Td}\cdot\left(\log^2(1/\varepsilon^2)\left(\frac{p}{1-p}\right)\mu\|\tilde{\mathbf{r}}\|_2^2\right).\end{aligned}$$

This proves the theorem.

5.7.2 Proof of Theorem 5.9

In this section we prove Theorem 5.9. Our proof of Theorem 5.9 relies on the following result from [36] which shows that any stochastic algorithm with a "good" estimator of the true gradient converges with rate $\mathcal{O}(\frac{1}{T})$. We translate this result to our setting.

Lemma 5.10 (Lemma 1 in [36]). *Suppose $\mathcal{L}$ is λ-strongly convex over a convex set $\mathcal{W}$, and that $\hat{\mathbf{g}}_t$ is an unbiased estimator of a subgradient $\mathbf{g}_t$ of the loss function $\mathcal{L}$ at $\mathbf{x}_t$, i.e., $\mathbb{E}\left[\hat{\mathbf{g}}_t\right] = \mathbf{g}_t$. Suppose also that for all t, $\mathbb{E}\left[\|\hat{\mathbf{g}}_t\|_2^2\right] \leq G$.*[7] *Then if we pick $\gamma_t = 1/\lambda t$, it holds for any T that*

$$\mathbb{E}\left[\|\mathbf{x}_T - \mathbf{x}^\star\|_2^2\right] \leq \frac{4G}{\lambda^2 T}.$$

Proof of Theorem 5.9. In order to apply Lemma 5.10, we need to show that the estimate of the gradient obtained by the master at each iteration is unbiased. To see this, recall that at each iteration t, the master computes the following estimate of the gradient:

$$\hat{\mathbf{g}}_t \triangleq \sum_{j=1}^{n}\sum_{i=1}^{m} \frac{\mathcal{I}_i^j}{d_i(1-p)} \nabla\mathcal{L}(\mathbf{a}_i, \mathbf{x}_t). \tag{5.19}$$

Therefore,

$$\mathbb{E}\left[\hat{\mathbf{g}}_t\right] = \sum_{j=1}^{n}\sum_{i=1}^{m} \frac{\mathbb{E}\mathcal{I}_i^j}{d_i(1-p)} \nabla\mathcal{L}(\mathbf{a}_i, \mathbf{x}_t). \tag{5.20}$$

Recall that $\mathcal{I}_i^j$ is an indicator function equal to 1 if worker j is non straggler and has data vector $\mathbf{a}_i$. Thus,

$$\begin{aligned}
\mathbb{E}\left[\hat{\mathbf{g}}_t\right] &= \sum_{j=1}^{n}\sum_{i=1}^{m} \frac{\mathbf{1}_{\text{worker } j \text{ has data vector } \mathbf{a}_i}}{d_i} \nabla\mathcal{L}(\mathbf{a}_i, \mathbf{x}_t) \\
&= \sum_{i=1}^{m} \nabla\mathcal{L}(\mathbf{a}_i, \mathbf{x}_t) \\
&= \nabla\mathcal{L}(A, \mathbf{x}_t).
\end{aligned}$$

Now, we need to show that under the conditions of the theorem, the variance $\mathbb{E}\|\hat{\boldsymbol{g}}(A, \mathbf{x}_t)\|_2^2$ is bounded. (Here, the randomness is over the choice of the stragglers in round t). As in

[7] Here, the randomness in the expectation is over the next round of stragglers, conditioned on the previous rounds.

the proof of Theorem 5.6, let Z_i be the binomial random variable that counts the number of non-stragglers (in a given round t) who have block i. Thus we have

$$\mathbb{E}\left[Z_i^2\right] = d_i p(1-p) + d_i^2(1-p)^2, \qquad \mathbb{E}\left[Z_{i_1} Z_{i_2}\right] = \frac{d_{i_1} d_{i_2}}{n} p(1-p) + d_{i_1} d_{i_2}(1-p)^2.$$

We compute

$$\begin{aligned}
\mathbb{E}\left[\|\hat{\boldsymbol{g}}(A, \mathbf{x}_t)\|_2^2\right] &\leq \mathbb{E}\left[\max_{\mathbf{x}\in\mathcal{W}} \|\hat{\boldsymbol{g}}(A, \mathbf{x})\|_2^2\right] \\
&= \mathbb{E}\left[\max_{\mathbf{x}\in\mathcal{W}} \left\|\sum_{j=1}^{n}\sum_{i=1}^{m} \frac{\mathcal{I}_i^j}{d_i(1-p)} \nabla\mathcal{L}(\mathbf{a}_i, \mathbf{x})\right\|_2^2\right] \\
&= \frac{1}{(1-p)^2}\mathbb{E}\left[\max_{\mathbf{x}\in\mathcal{W}} \left\|\sum_{i=1}^{m} \frac{Z_i}{d_i} \nabla\mathcal{L}(\mathbf{a}_i, \mathbf{x})\right\|_2^2\right] \\
&= \frac{1}{(1-p)^2}\mathbb{E}\left[\max_{\mathbf{x}\in\mathcal{W}} \sum_{i_1=1}^{m}\sum_{i_2=1}^{m} \frac{Z_{i_1} Z_{i_2}}{d_{i_1} d_{i_2}} \left\langle \nabla\mathcal{L}(\mathbf{a}_{i_1}, \mathbf{x}), \nabla\mathcal{L}(\mathbf{a}_{i_2}, \mathbf{x})\right\rangle\right] \\
&\leq \frac{1}{(1-p)^2} \sum_{i_1=1}^{m}\sum_{i_2=1}^{m} \frac{\mathbb{E}[Z_{i_1} Z_{i_2}]}{d_{i_1} d_{i_2}} \max_{\mathbf{x}\in\mathcal{W}} \left\langle \nabla\mathcal{L}(\mathbf{a}_{i_1}, \mathbf{x}), \nabla\mathcal{L}(\mathbf{a}_{i_2}, \mathbf{x})\right\rangle,
\end{aligned}$$

where above we have used the fact that the terms $\mathbb{E}[Z_{i_1} Z_{i_2}]/(d_{i_1} d_{i_2})$ are all positive to move the maximum inside the sum. We have

$$\left\langle \nabla\mathcal{L}(\mathbf{a}_{i_1}, \mathbf{x}), \nabla\mathcal{L}(\mathbf{a}_{i_2}, \mathbf{x})\right\rangle \leq \|\nabla\mathcal{L}(\mathbf{a}_{i_1}, \mathbf{x})\|_2 \|\nabla\mathcal{L}(\mathbf{a}_{i_2}, \mathbf{x})\|_2$$

by Cauchy-Shwarz, and thus

$$\max_{\mathbf{x}\in\mathcal{W}} \left\langle \nabla\mathcal{L}(\mathbf{a}_{i_1}, \mathbf{x}), \nabla\mathcal{L}(\mathbf{a}_{i_2}, \mathbf{x})\right\rangle \leq \max_{i\in[n]} \max_{\mathbf{x}\in\mathcal{W}} \|\nabla\mathcal{L}(\mathbf{a}_i, \mathbf{x})\|_2^2 \leq C^2,$$

by the assumptions of the theorem. Thus, we may continue the derivation above as

$$\begin{aligned}
\mathbb{E}\left[\|\hat{\boldsymbol{g}}(A,\mathbf{x}_t)\|_2^2\right] &\le \frac{C^2}{(1-p)^2}\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\frac{\mathbb{E}[Z_{i_1}Z_{i_2}]}{d_{i_1}d_{i_2}} \\
&= \frac{C^2}{(1-p)^2}\left(\sum_{i=1}^{m}\left(\frac{d_i p(1-p)+d_i^2(1-p)^2}{d_i^2}\right)\right) \\
&\quad + \frac{C^2}{(1-p)^2}\left(\sum_{i_1\neq i_2}\left(\frac{d_{i_1}d_{i_2}p(1-p)}{nd_{i_1}d_{i_2}}+\frac{d_{i_1}d_{i_2}(1-p)^2}{d_{i_1}d_{i_2}}\right)\right) \\
&= C^2\left(\sum_{i=1}^{m}\left(\frac{p}{(1-p)d_i}+1\right)+\sum_{i_1\neq i_2}\left(\frac{p}{n(1-p)}+1\right)\right) \\
&\le mC^2\left(\frac{p}{1-p}\cdot\frac{1}{d_{\min}}+\frac{(m-1)p}{n(1-p)}+m\right)
\end{aligned}$$

Plugging this estimate into Lemma 5.10 proves Theorem 5.9.

□

5.8 Related Work

In this section we survey the related work more broadly than in the introduction.

Coding techniques for straggler mitigation Straggler workers are the bottleneck of distributed systems and mitigation of stragglers is a must [5]. Amongst popular techniques, coding theoretic techniques are being used for straggler mitigation in different applications such as machine learning, see e.g. [10–13, 15, 100, 101, 103, 111, 129–131, 131–133, 133, 148–150], matrix multiplication, see e.g., [6,18,27,97,110,128,151–154], linear transforms, see e.g., [7,98,99,155], and content download, see e.g., [94–96, 156–158].

There is a growing body of work on *gradient coding,* which focuses on the special and important case of gradient descent. For example, [10–12] present coding techniques to avoid stragglers and perform a gradient descent update at each iteration, i.e., at each iteration the master observes the gradient evaluated at the whole data matrix A. In this framework, the master distributes the data to workers with redundancy. In the works cited above, the goal is to *exactly* compute the gradient, even in the presence of stragglers, and the amount of redundancy depends on the number of stragglers to be tolerated. However, it is still useful (and often much more efficient) to approximate the gradient, rather than computing it exactly. This setting is what our work focuses on and we discuss it in more detail below.

Approximate gradient coding In *approximate gradient coding,* the goal is not to compute the gradient exactly, but rather to compute an approximation to the gradient. This is the approach that we take, and there are several previous works which do this. The ones most relevant to our work are [13–17, 39–41, 142], which we discuss in more detail below.

In [39], the authors consider a setting where there are "extra" workers, and all the workers sample data randomly from B at each round. Again, the master waits for the fastest $n - s$ workers to complete. This is quite similar to the Ignore–Stragglers–SGD scheme; the difference is that in Ignore–Stragglers–SGD, the data is partitioned among the workers and the data held by the workers is fixed throughout the algorithm, while in [39] the workers sample a fresh subset of data at each iteration. The sampling of the data in [39] is done with replacement which may incur redundancy in the data held by the workers.

In [15], the authors focus on linear loss functions and use LDPC codes to encode the data sent to the workers. If fewer than s stragglers are present, then the master can compute the exact gradient. However, if more than s stragglers are present the master leverages the LDPC code to computes an estimate of the gradient. In [40] the authors propose a replication-based scheme to distribute the data to the workers. The data distribution scheme can be seen as a bipartite graph with the data vectors on one side and the workers on the other. An edge is drawn between a data vector i and a worker j if the vector $\mathbf{a}_i$ is given to worker j. The distribution of the degrees of the nodes corresponding to data vectors and to workers are drawn according to an LDGM scheme. The main drawback is that the master has to run a decoding algorithm to decode the sum of the partial gradients at each iteration.

In [13, 14], the authors present approximate gradient schemes. The main idea is to bound the distance between the computed approximate gradient and the actual gradient at each step. Both of these schemes have a similar framework to ours: the data is replicated among nodes according to an appropriate design, and the workers return a linear combination of the gradients that they can compute. In [13] the authors present a data replication scheme based on Ramanujan graphs. In [14] the authors present two constructions. The first is based on fractional repetition codes (FRC) and partitions the workers and data into blocks; within a block, each worker receives every data point from the corresponding block. The second construction called *Bernoulli Gradient Coding* (BGC) distributes each data point randomly to d different workers. We note that BGC is an approximation of the pairwise-balanced schemes we consider and can be seen as a case of SGC when all the data $\mathbf{a}_i$ have the same norm.

In [41] the authors present a data replication scheme based on balanced incomplete block designs. The scheme guarantees that the computed estimate of the gradient is close to the actual gradient even if the stragglers are not chosen randomly across iterations. This work is motivated by systems in which stragglers cannot be modeled statistically.

In [142], the authors also study a replication-based scheme, but they focus on a model where individual workers return many gradients asynchronously, rather than returning a linear combination of the gradients they can compute.

In [16] the authors present fundamental bounds on the error between the approximated gradient and true gradient at each round as function of the redundancy. In [17], the authors analyze the convergence rate of the fractional repetition scheme presented in [14] and show that under standard assumptions on the loss function, the algorithm maintains the convergence rate of centralized stochastic gradient descent.

Other work on stochastic gradient descent Beginning with its introduction in [159], there has been a huge body of work on stochastic gradient descent (in a setting without stragglers), and we draw on this mathematical framework for our theoretical results. In the special case of ℓ_2 loss (which we focus on in this work), SGD coincides with the randomized Kaczmarz method [144, 145], and our proof of Theorem 5.6 is inspired by these analyses.

There has been a great deal of work on SGD and Batch–SGD in distributed settings where there is no redundancy of the data between workers and dealing with stragglers is not the primary concern, see e.g., [146, 160–163]. In addition to the synchronous setting in which the master waits for all workers to make an update on $\mathbf{x}$, there has been work on the *asynchronous* setting in which the master makes an update on $\mathbf{x}$ every time a worker gets back, see e.g., [93, 140, 164–166]. For example, [166] shows that asynchronous SGD asymptotically behaves similarly to synchronous SGD in terms of convergence for convex optimization and under the similar assumptions on the loss function. In [165], the authors compare the convergence rate of synchronous and asynchronous SGD as a function of the wall clock time rather than number of iterations.

Chapter 6

Universally Robust Private Information Retrieval

We consider the problem of private information retrieval (PIR) as an extension to private coded computing. We follow the nomenclature of the PIR literature: the master is a *user* who wants to retrieve a file from a *database* owned by several *servers*. We consider the setting where the database consists of m files replicated on n servers that can possibly collude. We focus on devising robust PIR schemes that can tolerate stragglers, i.e., slow or unresponsive servers.

In many settings, the number of stragglers is not known a priori or may change with time. We define universally robust PIR as schemes that achieve asymptotic, in m, PIR capacity, i.e., asymptotically optimal download rate, simultaneously for any number of stragglers up to a given threshold. We introduce Staircase-PIR schemes and prove that they are universally robust. Towards that end, we establish an equivalence between robust PIR and communication efficient secret sharing.

6.1 Introduction

We consider the problem of designing PIR schemes on replicated data [43, 44], i.e., the data consisting of m files is *replicated* on n servers that can possibly collude. We use the terminology of the PIR literature: workers are referred to as servers, the master is a now a user who is querying the data. A user queries the servers to obtain a file of interest while keeping the identity of the file private, even if z servers collude with $z < n$. We focus on information-theoretic privacy (instead of computational privacy), which guarantees privacy without making any assumption on the computation power of the servers as long as no more than z servers collude.

Robustness: Under the setting described above, we are interested in PIR schemes that are robust. A robust PIR scheme allows the user to retrieve the file by receiving responses from any k servers $z < k \leq n$. The robustness property is motivated by the need to mitigate the effect of stragglers [5, 6, 10, 11, 26, 28, 76, 95–98, 149], i.e., slow or unresponsive servers.

PIR capacity: A challenge in designing PIR schemes is maximizing their download rate defined as the ratio of the retrieved file size to the amount of information downloaded by the user. The PIR capacity is defined as the maximal rate achievable by a PIR scheme and was shown in [46] to be

$$C_m(z,k) = \frac{1 - z/k}{1 - (z/k)^m}. \tag{6.1}$$

Note that $C_m(z,k)$ converges exponentially with the number of files m to its asymptotic value

$$C(z,k) = 1 - \frac{z}{k}. \tag{6.2}$$

For instance, for $m = 3$ files and $k = 10$ servers, $C_3(1,10)$ is at 99% of its asymptotic value $C(1,10)$ (assuming no collusion $z = 1$). Given the tremendous amount of files being stored in current systems, we focus on PIR schemes achieving the asymptotic capacity given in (6.2).

Universality: In many cases, the exact number of stragglers (slow or unresponsive servers) is not

	Server1	Server 2	Server 3
Storage	$\mathbf{x}$	$\mathbf{x}$	$\mathbf{x}$
Queries	$\mathbf{r}_1$	$\mathbf{e}_i + \mathbf{r}_1$	$2\mathbf{e}_i + \mathbf{e}_{m+i} + \mathbf{r}_1$
	$\mathbf{r}_2$	$\mathbf{e}_{m+i} + \mathbf{r}_2$	$2\mathbf{e}_{m+i} + \mathbf{r}_2$
Responses	$\mathbf{r}_1^T\mathbf{x}$	$(\mathbf{e}_i + \mathbf{r}_1)^T\mathbf{x}$	$(2\mathbf{e}_i + \mathbf{e}_{m+i} + \mathbf{r}_1)^T\mathbf{x}$
	$\mathbf{r}_2^T\mathbf{x}$	$(\mathbf{e}_{m+i} + \mathbf{r}_2)^T\mathbf{x}$	$(2\mathbf{e}_{m+i} + \mathbf{r}_2)^T\mathbf{x}$

Notation

$$\mathbf{x}^T = [x_1, \dots, x_m, x_{m+1}, \dots, x_{2m}]$$

$$f_i = [x_i,\ x_{m+i}] = [\mathbf{e}_i^T\mathbf{x},\ \mathbf{e}_{m+i}^T\mathbf{x}]$$

$$\mathbf{e}_i^T = [0, \dots, 0, \underbrace{1}_{i^{\text{th}}\ \text{entry}}, 0, \dots, 0]$$

$\mathbf{r}_1$ and $\mathbf{r}_2$: random vectors

Table 6.1: An example of Staircase-PIR code for $n = 3$, $k = 2$ and $z = 1$. The user sends 2 sub-queries to each server. Each server projects the data on the sub-query vectors and sends the result to the user. If all 3 servers are not stragglers, the user only downloads the first response (in blue) from each server to retrieve the required file as follows $x_i = (\mathbf{e}_i + \mathbf{r}_1)^T\mathbf{x} - \mathbf{r}_1^T\mathbf{x}$ and $x_{m+i} = (2\mathbf{e}_i + \mathbf{e}_{m+i} + \mathbf{r}_1)^T\mathbf{x} - \mathbf{r}_1^T\mathbf{x} - 2x_i$. The rate of this scheme is equal to $C(1,3) = 2/3$, because the user downloads 3 responses to retrieve the 2 parts of the file $f_i = [x_i,\ x_{m+i}]$. If 1 server is a straggler, the user downloads 2 responses from each of the remaining servers to retrieve the file. The user can retrieve the file irrespective of which server is a straggler. The rate of this scheme is $C(1,2) = 1/2$, because the user downloads 4 responses to retrieve the file.

known a priori. This motivates the study of universally robust PIR schemes that can tolerate a varying number of stragglers. More precisely, a universally robust PIR scheme is a scheme that allows the user to retrieve the file from any d servers, $k \leq d \leq n$, while achieving the asymptotic PIR capacity $C(z,d)$ given in (6.2) simultaneously for all possible values of d. While a robust PIR scheme designed for the worst case number of stragglers ($d = k$) can tolerate up to $n - k$ stragglers, it does not necessarily achieve the capacity for all possible values of d.

In the case where the stragglers are non-responsive, universally robust PIR guarantees that the user can always obtain the file even if up to $n - k$ servers do not reply to its queries. On the other hand, when the stragglers are slow servers, universally robust PIR offers the user a tradeoff between download rate and waiting time. The user can pick the number of servers to wait for and universally robust PIR scheme achieves asymptotic capacity irrespective of this choice. We illustrate the idea in Example 6.1.

Example 6.1 (Universally robust Staircase-PIR)**.** *Let $n = 3$ servers and we want to tolerate up to 1 straggler, i.e., $k = 2$. Assume the servers do not collude, i.e., $z = 1$. We describe the Staircase-PIR that simultaneously achieves $C(1,2) = 1/2$ in the case of 1 straggler $d = 2$ and $C(1,3) = 2/3$ in the case of no stragglers $d = 3$.*

Let $f_1, \dots, f_m$ *be the files stored on the servers. We divide each file* f_i *into 2 parts and denote the* i^{th} *file as* $f_i = [x_i,\ x_{m+i}]$. *We represent the data by the vector*[1] $\mathbf{x}$ *where the entries of this vector are indexed as follows* $\mathbf{x} = [x_1, \dots, x_m, x_{m+1}, \dots, x_{2m}]^T$. *Let* $\mathbf{e}_i$ *denote the all zero vector of length* $2m$ *with a '1' in the* i^{th} *entry. The* i^{th} *file can be expressed as* $f_i = [\mathbf{e}_i^T\mathbf{x},\ \mathbf{e}_{m+i}^T\mathbf{x}]$ *which is the projection of* $\mathbf{x}$ *on* $\mathbf{e}_i$ *and* $\mathbf{e}_{m+i}$. *To construct the Staircase-PIR scheme, we use two independent random vectors* $\mathbf{r}_1$ *and* $\mathbf{r}_2$ *each of length* $2m$ *and with entries drawn uniformly at random from* $\mathbb{F}_2$. *We encode the queries using Staircase codes as shown in Table 6.1.*

The user sends 2 *sub-queries to each server which projects the data on the queries and sends the result to the user. If one server is straggler, the user downloads all the sub-queries from the other* 2 *servers to retrieve the file. For instance, if server* 3 *is the straggler, the user downloads all the responses from servers* 1 *and* 2, 4 *responses in total, and retrieves* $f_i = [x_i,\ x_{m+i}]$ *as* $x_i = (\mathbf{e}_i + \mathbf{r}_1)^T\mathbf{x} - \mathbf{r}_1^T\mathbf{x}$ *and* $x_{m+i} = (\mathbf{e}_{m+i} + \mathbf{r}_2)^T\mathbf{x} - \mathbf{r}_2^T\mathbf{x}$. *The rate in this case is equal to* $2/4 = 1/2 = C(1,2)$. *However, if no server is a straggler, the user only downloads the first response (in blue) of all* 3 *servers to retrieve the file. The rate in this case is equal to* $2/3 = C(1,3)$. *Privacy is achieved because* $\mathbf{e}_i$ *and* $\mathbf{e}_{m+i}$ *are padded with random vectors* $\mathbf{r}_1$, $\mathbf{r}_2$.

Related work: Private information retrieval was introduced by Chor et al. [43,44] and was followed up by a large body of work, e.g., [46,47,51,167–175]. The literature mainly focused on reducing the communication cost of privately retrieving the file. The early body of work measured the communication cost by the amount of information uploaded (queries) and downloaded (responses) by the user [167–170]. Given the increasing size of stored files, the recent body of work measures the communication cost by the amount of information downloaded by the user, assuming the queries are too small compared to the downloaded files [46,47,51,171–175] which is the assumption we adopt in this paper.

Robust PIR was studied in the literature, e.g., [45–52] and the capacity of robust PIR schemes under download cost was characterized in [46,47,51] for replicated data. The common focus of the literature has been on designing robust PIR that are not necessarily universal,

[1]The superscript T denotes the transpose operator.

which are tailored to a specific number of stragglers. In [49] the authors present a universally robust PIR scheme for the no collusion case and when the data is stored on the servers using a maximum distance separable (MDS) code.

Contributions: We introduce Staircase-PIR, a universally robust PIR scheme achieving asymptotic capacity for any number of stragglers up to a given threshold. Compared to the previous work on universal PIR [49], this work allows servers' collusion but is restricted to the case of replicated data. The main ingredient of the proposed scheme is Staircase secret sharing codes introduced by the authors in [24,25]. Moreover, we establish an equivalence between robust PIR schemes achieving asymptotic capacity and communication efficient secret sharing schemes.

6.2 Problem formulation and main results

We consider robust private information retrieval. The data $\mathbf{x}$ is formed of m files $f_1, \dots, f_m$ and is replicated on n servers. A user wants to retrieve a file f_i from the data without revealing the identity i of the file to the servers. A robust PIR scheme encodes a set of queries $\mathbf{q}_1, \dots, \mathbf{q}_n$ to be sent by the user to all the servers. Let $\mathtt{W}$ denote the random variable representing the identity of the file that the user wants and let $\mathtt{F}$ be the random variable representing the file f_i. Let $\mathtt{Q}_i$ denote the random variable representing query $\mathbf{q}_i$ and let $[n] \triangleq \{1, \dots, n\}$. For any subset $\mathcal{B} \subseteq [n]$ denote by $\mathtt{Q}_{\mathcal{B}}$ the set of random variables representing the queries indexed by $\mathcal{B}$, i.e., $\mathtt{Q}_{\mathcal{B}} = \{\mathtt{Q}_i : i \in \mathcal{B}\}$. Let $H(.)$ denote the entropy function. Then, a universally optimal robust PIR scheme is defined as follows.

Definition 6.1 (Universally robust PIR)**.** A universally robust PIR (UR-PIR) scheme is a scheme that satisfies the following properties:

1) Privacy: Any subset of z or less queries should not reveal any information about the identity of the file, i.e.,

$$H(\mathtt{W} \mid \mathtt{Q}_{\mathcal{Z}}) = H(\mathtt{W}), \forall \mathcal{Z} \subset [n] \text{ s.t. } |\mathcal{Z}| = z. \tag{6.3}$$

2) Robustness: When receiving the responses of d servers, $k \leq d \leq n$, the user obtains the file,

$$H(\mathtt{F} \mid \mathtt{Q}_{\mathcal{A}}) = 0, \forall \mathcal{A} \subseteq [n] \text{ s.t. } k \leq |\mathcal{A}| \leq n. \tag{6.4}$$

3) Optimality: The capacity of robust PIR is [46]

$$C(z,k) = 1 - \frac{z}{k}. \tag{6.5}$$

We say that the scheme is optimal if the rate of the scheme achieves $C(z,k)$ given in (6.5), i.e.,

$$\frac{H(\mathtt{F})}{H(\mathtt{Q}_{\mathcal{A}})} = 1 - \frac{z}{d}, \forall \mathcal{A} \subset [n] \text{ s.t. } |\mathcal{A}| = d, \tag{6.6}$$

for all d, $k \leq d \leq n$.

In addition, we refer to a robust PIR achieving capacity $C(z,k)$ as an (n,k,z) robust PIR and we refer to a universally robust PIR achieving capacity $C(z,d)$ for all $k \leq d \leq n$ as an (n,k,z) UR-PIR. We introduce Staircase-PIR, a deterministic construction for all (n,k,z) UR-PIR schemes.

Theorem 6.2. *The (n,k,z) Staircase-PIR scheme described in Section 6.3.1 is a universally robust PIR, i.e., satisfies the required privacy and robustness constraints given in (6.3) and (6.4) for any given $z < k \leq n$, and achieves the asymptotic capacity of robust PIR*

$$C(z,d) = 1 - \frac{z}{d},$$

simultaneously for all d such that $k \leq d \leq n$.

Symbol	Meaning	Symbol	Meaning
n	number of servers	$C_m(n,k)$	capacity of PIR scheme
k	threshold on non stragglers	$C(n,k)$	asymptotic capacity of PIR scheme
z	number of colluding servers	E	matrix containing $\mathbf{e}_i$'s
d	number of non stragglers	R_i	matrix containing $\mathbf{r}_i$'s
m	total number of files in $\mathbf{x}$	D_j	matrix with entries from E and R_i, $i \leq j$
f_i	i^{th} file of the data $\mathbf{x}$	M	structured matrix containing E, R_i's and D_j's
d_i	defined as $d_i \triangleq n-i+1$	$\mathbf{x}$	the data stored at the servers
α_i	defined as $\alpha \triangleq d_i - z$	$\mathbf{q}_i$	query sent to server i
α	defined as $\alpha \triangleq \text{LCM}(\alpha_1, \ldots, \alpha_{n-k})$	$\mathbf{r}$	random vector
α'	defined as $\alpha' \triangleq (k-z)\alpha$	$\mathbf{0}$	matrix with all 0 entries
$\mathbf{e}_i$	all zero vector with 1 in the i^{th} position	$\mathbf{e}'_{j,i}$	defined as $\mathbf{e}'_{j,i} \triangleq \mathbf{e}_{(j-1)m+i}$

Table 6.2: Summary of notations for this Chapter.

6.3 Staircase-PIR scheme

6.3.1 Staircase-PIR construction

We describe the (n,k,z) Staircase-PIR scheme. The scheme consists of three steps: i) the user encodes the queries $\mathbf{q}_1, \ldots, \mathbf{q}_n$ and sends them to the servers; ii) each server j projects the data on the received query, i.e., computes $\mathbf{q}_j^T\mathbf{x}$ and sends the result to the user; and iii) the user decodes the requested file. We start by explaining the encoding of the queries. Let $d_j = n-j+1$, and $\alpha_j = d_j - z$, $j = 1, \ldots, n-k+1$. Staircase-PIR divides each query into α sub-queries, where $\alpha = LCM(\alpha_1, \ldots, \alpha_{n-k})$ is the least common multiple of all the α_j's except for the last $\alpha_{n-k+1} = k-z$. Consequently, the construction assumes that each file f_i of the data is divided into $\alpha' = (k-z)\alpha$ parts, i.e., $f_i = [x_i, x_{m+i}, \ldots, x_{(\alpha'-1)m+i}] = [\mathbf{e}_i^T\mathbf{x}, \ldots, \mathbf{e}_{(\alpha'-1)m+i}^T\mathbf{x}]$, where $\mathbf{x} = [x_1, \ldots, x_m, x_{m+1}, \ldots, x_{\alpha' m}]^T$. For simplicity of notation, let $\mathbf{e}'_{j,i} \triangleq \mathbf{e}_{(j-1)m+i}$ where i is the index of the requested file and is dropped when it is clear from the context.

To retrieve file i, the user encodes $\mathbf{e}'_1, \ldots, \mathbf{e}'_{\alpha'}$ together with $z\alpha$ iid random vectors, drawn uniformly at random from $\mathbb{F}_2$ and independently of the $\mathbf{e}'_j$'s. To encode the queries, we arrange the $\mathbf{e}'_j$'s in an $\alpha_1 \times \alpha'/\alpha_1$ matrix E and the random vectors in $n-k+1$ matrices $R_j, j = 1, \ldots, n-k+1$, of respective dimensions $z \times \alpha'/\alpha_j\alpha_{j-1}$ (take $\alpha_0 = 1$). Note that each entry of the matrices E and the R_j's is a vector of dimension αm.

We arrange E and the R_j's in $n-k+1$ matrices M_j, $j = 1, \dots, n-k+1$, as follows,

$$M_1 = \begin{bmatrix} E \\ \\ R_1 \end{bmatrix} \quad \text{and } M_j = \begin{bmatrix} D_{j-1} \\ R_j \\ \mathbf{0} \end{bmatrix} \quad j \neq 1,$$

where, D_j is a matrix of dimensions $\alpha_{j+1} \times \alpha'/\alpha_j\alpha_{j+1}$, $j = 1, \dots, n-k$, formed of the $(n-j+1)^{th}$ row of $\begin{bmatrix} M_1 \; M_2 \; \cdots \; M_j \end{bmatrix}$ wrapped around to fit the above mentioned dimensions. Each M_j matrix is completed to n rows with the all zero matrix $\mathbf{0}$. We obtain the encoding matrix M defined in Table 6.3 by concatenating the $n-k+1$ matrices M_j, $j = 1, \dots, n-k+1$. Note that the positions of R_j and D_{j-1} (similarly R_1 and E) in M can be switched without affecting the construction.

$$M = \left[\begin{array}{c|c|c|c|c} & & D_2 & \cdots & D_{h-1} \\ & D_1 & & & R_h \\ E & & R_3 & \cdots & \\ & R_2 & & & \mathbf{0} \\ R_1 & \mathbf{0} & \mathbf{0} & \cdots & \end{array}\right].$$

Table 6.3: The structure of the matrix M used to encode the queries of Staircase-PIR.

Encoding of the queries: Let V be an $n \times n$ Vandermonde[2] matrix over $\mathbb{F}_q$, $q > n$. The matrix M is multiplied by V to obtain the query matrix $Q = VM$. The queries sent to the servers are the n rows of Q. Note that each row of Q consists of α entries, hence each query is divided in α sub-queries.

Retrieving the file: To retrieve the wanted file by waiting for any d_j servers indexed by $\mathcal{J} \subseteq [n]$, the user only downloads the projection of $\mathbf{x}$ on the first α'/α_j sub-queries from each contacted server corresponding to $\left(\mathbf{v}_l \begin{bmatrix} M_1 \; \cdots \; M_j \end{bmatrix}\right)^T \mathbf{x}$, for all $l \in \mathcal{J}$, where $\mathbf{v}_l$ denote the l^{th} row of V. Decoding all the $\mathbf{e}'$'s from the received parts of the responses is guaranteed by Theorem 2.2, therefore retrieving the file $[\mathbf{e}'_1, \dots, \mathbf{e}'_{\alpha'}]^T\mathbf{x}$ follows from the linearity of the scheme.

Optimality: When waiting for d_j servers, the user downloads $d_j\alpha'/\alpha_j$ responses to retrieve the α' parts of the file. Therefore, the rate of the scheme is given by $\alpha'/(d_j\alpha'/\alpha_j) = \alpha_j/d_j =$

[2]All square sub-matrices formed by consecutive columns of V must be invertible. Two family of matrices satisfying this property are Vandermonde and Cauchy matrices.

$(d_j - z)/d_j = C(z, d_j)$ as given in (6.2).

Privacy: Each subset of at most z servers obtain no information about the identity of the wanted file. The servers only observe the queries, Theorem 2.2 guarantees that the queries sent to any z servres leak no information about the $\mathbf{e}'_j$'s. We skip this proof and refer the interested reader to Section 2.5 for a detailed proof.

6.3.2 Examples of Staircase-PIR

First we show how the scheme in Example 6.1 was obtained using the general construction.

Example 6.1 (Continued)**.** *Recall that we want to construct an $(n, k, z) = (3, 2, 1)$ Staircase-PIR scheme. Each file is divided into two parts $f_i = [\mathbf{e}_i^T\mathbf{x},\ \mathbf{e}_{m+i}^T\mathbf{x}]$ and the construction uses two random vectors $\mathbf{r}_1$ and $\mathbf{r}_2$. The matrix M is created by arranging the vectors $\mathbf{e}_i$ and $\mathbf{e}_{m+i}$ in $E = [\mathbf{e}_i,\ \mathbf{e}_{m+i}]$ and by having $R_1 = [\mathbf{r}_1]$, $R_2 = [\mathbf{r}_2]$ and $D_1 = [\mathbf{e}_{m+i}]$. The matrix M in this example is given by*

$$M = \begin{bmatrix} R_1 & R_2 \\ & D_1 \\ E & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \\ \mathbf{e}_i & \mathbf{e}_{m+i} \\ \mathbf{e}_{m+1} & \mathbf{0} \end{bmatrix}.$$

The user constructs the query matrix $Q = VM$ as given in (6.7), where each row $\mathbf{q}_i$ of Q is the query sent to server i. The queries are given in Table 6.1.

$$Q = VM = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \\ \mathbf{e}_i & \mathbf{e}_{m+i} \\ \mathbf{e}_{m+1} & \mathbf{0} \end{bmatrix}. \tag{6.7}$$

Next we give a second example that illustrates in more details the general construction of Staircase-PIR.

Example 6.2. *We construct an $(n, k, z) = (4, 2, 1)$ Staircase-PIR. Let $d_1 = 4,\ d_2 = 3,\ d_3 = 2$, $\alpha_1 = 3$, $\alpha_2 = 2$, $\alpha_3 = 1$ and $\alpha = LCM(\alpha_1, \alpha_2) = LCM(3, 2) = 6$. We divide the files into $\alpha' = (k - z)\alpha = 6$ parts each, i.e., the file f_i is formed of $f_i = [\mathbf{e}_i^T\mathbf{x}, \dots, \mathbf{e}_{5m+i}^T\mathbf{x}] \triangleq [\mathbf{e}'^T_1\mathbf{x}, \dots, \mathbf{e}'^T_6\mathbf{x}]$. The construction uses $z\alpha = 6$ iid random vectors drawn uniformly at random from $\mathbb{F}_2$. The*

Server 1	Server 2
$(\mathbf{e}'_1+\mathbf{e}'_2+\mathbf{e}'_3+\mathbf{r}_1)^T\mathbf{x}$	$(\mathbf{e}'_1+2\mathbf{e}'_2+4\mathbf{e}'_3+3\mathbf{r}_1)^T\mathbf{x}$
$(\mathbf{e}'_4+\mathbf{e}'_3+\mathbf{e}'_6+\mathbf{r}_2)^T\mathbf{x}$	$(\mathbf{e}'_4+2\mathbf{e}'_5+4\mathbf{e}'_6+3\mathbf{r}_2)^T\mathbf{x}$
$(\mathbf{r}_1+\mathbf{r}_2+\mathbf{r}_3)^T\mathbf{x}$	$(\mathbf{r}_1+2\mathbf{r}_2+4\mathbf{r}_3)^T\mathbf{x}$
$(\mathbf{e}'_3+\mathbf{r}_4)^T\mathbf{x}$	$(\mathbf{e}'_3+2\mathbf{r}_4)^T\mathbf{x}$
$(\mathbf{e}'_6+\mathbf{r}_5)^T\mathbf{x}$	$(\mathbf{e}'_6+2\mathbf{r}_5)^T\mathbf{x}$
$(\mathbf{r}_3+\mathbf{r}_6)^T\mathbf{x}$	$(\mathbf{r}_3+2\mathbf{r}_6)^T\mathbf{x}$

Server 3	Server 4
$(\mathbf{e}'_1+3\mathbf{e}'_2+4\mathbf{e}'_3+2\mathbf{r}_1)^T\mathbf{x}$	$(\mathbf{e}'_1+4\mathbf{e}'_2+\mathbf{e}'_3+4\mathbf{r}_1)^T\mathbf{x}$
$(\mathbf{e}'_4+3\mathbf{e}'_5+4\mathbf{e}'_6+2\mathbf{r}_2)^T\mathbf{x}$	$(\mathbf{e}'_4+4\mathbf{e}'_5+\mathbf{e}'_6+4\mathbf{r}_2)^T\mathbf{x}$
$(\mathbf{r}_1+3\mathbf{r}_2+4\mathbf{r}_3)^T\mathbf{x}$	$(\mathbf{r}_1+4\mathbf{r}_2+\mathbf{r}_3)^T\mathbf{x}$
$(\mathbf{e}'_3+3\mathbf{r}_4)^T\mathbf{x}$	$(\mathbf{e}'_3+4\mathbf{r}_4)^T\mathbf{x}$
$(\mathbf{e}'_6+3\mathbf{r}_5)^T\mathbf{x}$	$(\mathbf{e}'_6+4\mathbf{r}_5)^T\mathbf{x}$
$(\mathbf{r}_3+3\mathbf{r}_6)^T\mathbf{x}$	$(\mathbf{r}_3+4\mathbf{r}_6)^T\mathbf{x}$

Table 6.4: The responses sent by the servers when using an $(n, k, z) = (4, 2, 1)$ Staircase-PIR scheme.

$\mathbf{e}'_j$*'s and the random vectors are arranged in the following matrices,*

$$E = \begin{bmatrix} \mathbf{e}'_1 & \mathbf{e}'_4 \\ \mathbf{e}'_2 & \mathbf{e}'_5 \\ \mathbf{e}'_3 & \mathbf{e}'_6 \end{bmatrix}, \; R_1 = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{bmatrix},$$

$$R_2 = \begin{bmatrix} \mathbf{r}_3 \end{bmatrix}, \; \textit{and } R_3 = \begin{bmatrix} \mathbf{r}_4 & \mathbf{r}_5 & \mathbf{r}_6 \end{bmatrix}.$$

To build the matrix M which will be used for encoding the queries, we start with

$$M_1 = \begin{bmatrix} \mathbf{e}'_1 & \mathbf{e}'_2 & \mathbf{e}'_3 & \mathbf{r}_1 \\ \mathbf{e}'_4 & \mathbf{e}'_5 & \mathbf{e}'_6 & \mathbf{r}_2 \end{bmatrix}^T.$$

Then, D_1 is the $\alpha_2 \times \alpha'/\alpha_1\alpha_2 = 2 \times 1$ matrix containing the entries of the n^{th} row of M_1, i.e., $D_1 = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{bmatrix}^T$. Therefore, $M_2 = \begin{bmatrix} D_1 & R_2 & \mathbf{0} \end{bmatrix}^T = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & 0 \end{bmatrix}^T$. Similarly, we have $D_2 = \begin{bmatrix} \mathbf{e}'_3 & \mathbf{e}'_6 & \mathbf{r}_3 \end{bmatrix}$ and

$$M_3 = \begin{bmatrix} \mathbf{e}'_3 & \mathbf{e}'_6 & \mathbf{r}_3 \\ \mathbf{r}_4 & \mathbf{r}_5 & \mathbf{r}_6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

We obtain M by concatenating M_1, M_2 and M_3,

$$M = \begin{bmatrix} \mathbf{e}'_1 & \mathbf{e}'_4 & \mathbf{r}_1 & \mathbf{e}'_3 & \mathbf{e}'_6 & \mathbf{r}_3 \\ \mathbf{e}'_2 & \mathbf{e}'_5 & \mathbf{r}_2 & \mathbf{r}_4 & \mathbf{r}_5 & \mathbf{r}_6 \\ \mathbf{e}'_3 & \mathbf{e}'_6 & \mathbf{r}_3 & 0 & 0 & 0 \\ \mathbf{r}_1 & \mathbf{r}_2 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The matrix V is the $n \times n = 4 \times 4$ Vandermonde matrix over $\mathbb{F}_5$ given in (6.8). The query $\mathbf{q}_j$ sent to server j is the j^{th} row of the matrix $Q = VM$. The responses to the queries, i.e., $Q^T\mathbf{x}$ are given in Table 6.4.

$$V = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \end{bmatrix}. \tag{6.8}$$

After receiving the query $\mathbf{q}_j$, each server j projects the data $\mathbf{x}$ on $\mathbf{q}_j$ and sends the result back to the user. We illustrate how the user can retrieve the wanted file by achieving the PIR capacity $C(z, d)$ given in (6.2) simultaneously for $d_1 = 4$, $d_2 = 3$ and $d_2 = 2$.

Suppose the user waits for $d_1 = 4$ servers. The user downloads the first $\alpha'/\alpha_1 = 2$ responses of each server corresponding to $VM_1^T\mathbf{x}$ (first two rows in black in Table 6.4). Recall that V is a Vandermonde matrix, hence is invertible. The user multiplies the received responses by the inverse of V to decode $M_1^T\mathbf{x}$ which contains ${\mathbf{e}'_j}^T\mathbf{x}$ for $j = 1, \dots, 6$, therefore retrieving the desired file. The rate of this PIR scheme is equal to $6/8 = 3/4 = C(1, 4)$, because the user decodes the 6 parts of the file by downloading 8 responses.

If the user waits for $d_2 = 3$ servers, the user downloads the first $\alpha'/\alpha_2 = 3$ responses of each contacted server corresponding to $V_{\mathcal{J}} \begin{bmatrix} M_1 & M_2 \end{bmatrix}^T \mathbf{x}$ (in black and red), where $V_{\mathcal{J}}$ is the matrix formed by the rows of V indexed by $\mathcal{J} \subset [n]$. Recall that $V_{\mathcal{J}}$ here is a 3×4 Vandermonde matrix. The user can retrieve the file as follows. Since M_2 has a 0 as its last entry, $V_{\mathcal{J}}M_2^T\mathbf{x}$ reduces to $V'_{\mathcal{J}}M_2^T\mathbf{x}$, where $V'_{\mathcal{J}}$ is the 3×3 invertible Vandermonde matrix formed of the first three columns of $V_{\mathcal{J}}$. The user can then decode $M_2^T\mathbf{x}$ which consists of $\mathbf{r}_1^T\mathbf{x}$, $\mathbf{r}_2^T\mathbf{x}$ and $\mathbf{r}_3^T\mathbf{x}$. By subtracting $\mathbf{r}_i^T\mathbf{x}$ from $V_{\mathcal{J}}M_1^T\mathbf{x}$, the user obtains $V'_{\mathcal{J}}{M'_1}^T\mathbf{x}$ where $M'_{\mathcal{J}}$ is the matrix formed of the the first three columns of M_1. By inverting $V'_{\mathcal{J}}$ the user can decode ${M'_1}^T\mathbf{x}$ which contains *the required file. The rate of this PIR scheme is equal to $6/9 = 2/3 = C(1,3)$, because the user decodes the 6 parts of file by downloading 9 responses.*

Following a similar procedure, the user can retrieve the file by downloading all the responses of any 2 servers. The rate here is $C(1,2) = 1/2$ because the user downloads 12 responses to decode the requested file.

On a high level, privacy is guaranteed because each sub-query is padded with a different random vector.

6.4 From secret sharing to PIR

The connection between secret sharing and PIR has been studied in the literature, e.g., [45,167,176]. Our Staircase-PIR construction was obtained using Staircase codes for communication efficient secret sharing explained in Chapter 2. In this section, we explore more this connection between communication efficient secret sharing [22–24, 64, 177] and capacity achieving robust PIR schemes. We redefine secret sharing and communication efficient secret sharing in the language of this chapter.

Definition 6.3 (Secret sharing scheme)**.** A secret sharing scheme is an encoding of a secret $\mathbf{s}$ into n shares $\mathbf{w}_1, \dots, \mathbf{w}_n$, stored on n servers, such that a user accessing any subset of z or less shares obtains no information about $\mathbf{s}$, however by accessing any collection of k or more shares the user can reconstruct the whole secret. Let $\mathtt{S}$ denote the random variable representing the secret $\mathbf{s}$ and $\mathtt{W}_i$ denote the random variable representing the share $\mathbf{w}_i$. A secret sharing scheme satisfies the following properties:

1) Perfect secrecy: expressed as

$$H(\mathtt{S} \mid \mathtt{W}_{\mathcal{Z}}) = H(\mathtt{S}), \forall \mathcal{Z} \subset [n] \text{ s.t. } |\mathcal{Z}| = z. \tag{6.9}$$

2) MDS: or reconstruction of the secret,

$$H(\mathtt{S} \mid \mathtt{W}_{\mathcal{A}}) = 0, \forall \mathcal{A} \subseteq [n] \text{ s.t. } |\mathcal{A}| = k, \tag{6.10}$$

and the secret is of size $(k - z)$ units as implied by (6.9) and (6.10) (see [23, Proposition 1]).

We refer to a secret sharing scheme as defined above as an (n, k, z) secret sharing.

Definition 6.4 (Communication efficient secret sharing)**.** A communication efficient secret sharing is a secret sharing scheme that allows the user to reconstruct the secret by downloading a part of any d shares, $k \leq d \leq n$. When accessing d servers, the user needs only to download the optimal rate $d(k - z)/(d - z)$ units of information given in [22, 23].

Now we are ready to describe how to obtain a robust PIR scheme from a secret sharing (SS) scheme. We call this construction SS-PIR construction.

SS-PIR construction: An (n, k, z) robust PIR can be constructed using linear secret sharing as follows. Let f_i, the i^{th} entry of the data $\mathbf{x}$, be the file of interest expressed as $f_i = x_i = \mathbf{e}_i^T\mathbf{x}$. To construct an (n, k, z) robust PIR scheme, the user encodes the queries $\mathbf{q}_1, \dots, \mathbf{q}_n$ using an (n, k, z) *linear* secret sharing scheme with $\mathbf{s} = \mathbf{e}_i$ and sends those queries to the n servers. After receiving the query, each server j projects the data $\mathbf{x}$ on $\mathbf{q}_j$ and sends $\mathbf{q}_j^T\mathbf{x}$ to the user.

Proposition 6.5. *An information retrieval scheme obtained from the SS-PIR construction is a robust PIR scheme, i.e., guarantees privacy and robustness and achieves asymptotic capacity equal to*

$$C(z, k) = 1 - \frac{z}{k}.$$

Proof. We show that an information retrieval scheme constructed using linear secret sharing scheme is a robust PIR scheme, i.e., guarantees privacy and robustness, that achieves asymptotic PIR capacity.

Privacy: Any subset of z or less servers obtain no information about the identity of the file of interest. Each server observes a query encoded using an (n, k, z) secret sharing. From the secrecy constraint of secret sharing (6.9), any z or less servers obtain no information about $\mathbf{e}_i$, which represents the identity of the file.

Robustness: After receiving the responses $\mathbf{q}_j^T\mathbf{x}$ from any k servers, $j \in [n]$, the user can retrieve f_i. From the MDS property of the secret sharing (6.10), the user can decode $\mathbf{e}_i$ from any k queries $\mathbf{q}_j$, $j \in [n]$. By linearity of the secret sharing scheme, after receiving k responses $\mathbf{q}_j^T\mathbf{x}$ from the servers, the user is able to decode $f_i = \mathbf{e}_i^T\mathbf{x}$. In other words, since all the queries are multiplied by the same vector $\mathbf{x}$, being able to decode the secret $\mathbf{s} = \mathbf{e}_i$ from the queries implies the ability of decoding the file $\mathbf{e}_i^T\mathbf{x}$ from the responses, c.f. Section 6.3.2.

Optimality: We show that this PIR scheme is optimal, i.e., achieves $C(z, k)$ given in (6.2). To retrieve the file, the user has to download k responses, i.e., k units of information. Secret sharing assumes that the size of the retrieved file is $k - z$ units of information[3]. Therefore, the rate of this PIR scheme is $(k - z)/k$ achieving $C(z, k)$. $\square$

We generalize Proposition 6.5 to show that a PIR scheme constructed using a communication efficient secret sharing is universally robust.

Proposition 6.6. *An information retrieval scheme constructed using an (n, k, z) linear communication efficient secret sharing scheme is a universally robust PIR that achieves PIR capacity*

$$C(z, d) = 1 - \frac{z}{d},$$

simultaneously for all $k \leq d \leq n$.

The implication of this Proposition is that any communication efficient secret sharing including Staircase codes can be used to obtain universally robust PIR through SS-PIR construction.

[3]The scheme can be scaled so that the file is of size 1 unit of information, each query becomes of size $1/(k-z)$.

Proof. We show that an information retrieval scheme encoding the queries using a linear communication efficient secret sharing scheme is an (n, k, z) UR-PIR. Note that privacy and robustness to any number of unresponsive servers are guaranteed by the properties of secret sharing and by the ability of decoding the secret by accessing any d shares. The additional property that we comment on is the optimality of retrieving the file when any number of servers d, $k \leq d \leq n$, are stragglers. When a user receives responses from d servers it only needs to download $d(k-z)/(d-z)$ units of information to retrieve the file of size $k-z$. Therefore, the rate of this scheme is equal to $(d-z)/d$ achieving (6.2) with equality. □

6.5 Conclusion

We study robust private information retrieval (PIR). A user wants retrieve a file by querying n servers without revealing the identity of the required file to the servers. We assume the servers can collude and consider the setting in which the servers might be stragglers, i.e., slow or unresponsive. We introduce Staircase-PIR, a universally robust PIR scheme that allows the user to successfully retrieve the file by waiting only for the non straggler servers. This scheme achieves the PIR capacity simultaneously for any number of stragglers up to a given threshold. Moreover, we give a general construction to obtain universally robust PIR from communication efficient secret sharing.

Chapter 7

Conclusion

This dissertation tackles the problem of private coded computation in the presence of stragglers. We focus on linear computation, i.e., matrix-vector multiplication. Private coded computation schemes based on classical secret sharing tolerate the presence of a fixed number of stragglers and guarantee privacy. Classical secret sharing schemes allow the master to share a secret with n workers such that any $k \leq n$ workers can decode the secret and any $z < k$ workers obtain no information about the secret, in an information theoretic sense.

We propose the use of communication efficient secret sharing in the setting where the workers have similar and fixed resources. In addition to the properties of secret sharing, a communication efficient secret sharing allows a legitimate user to decode the secret by contacting $k \leq d \leq n$ workers and downloading the minimum amount of information as a function of d to decode the secret. We construct Staircase codes, a new family of communication efficient secret sharing schemes, in Chapter 2 and analyze their performance in this private coded computation setting in Chapter 3.

In the setting where the workers have different and are time-varying resources, we propose new private and rateless codes called PRAC that adapt to the dynamically changing resources of the workers (Chapter 4). PRAC estimates the available resources at the workers and allocates tasks that are proportional to the estimated resources. We show that, in this setting, PRAC outperforms any scheme that pre-allocates the tasks to the workers.

We consider the extension of private coded computation to private information retrieval (PIR) and construct Staircase-PIR (Chapter 6) that achieves the minimum communication cost simultaneously for any number of stragglers, up to a given threshold.

Beyond linear coded computation, this dissertation contributes to the growing literature on approximate gradient coding. We introduce in Chapter 5 a new family of codes called stochastic gradient codes (SGC) for distributed gradient-descent-like algorithm for any additively separable loss function. SGC requires a constant amount of redundancy in distributing the data, tolerates any number of stragglers and enjoys a fast convergence rate.

In addition to the theoretical contributions, this dissertation provides extensive simulations on MATLAB and extensive implementation of the proposed codes on Amazon EC2 clusters and Android devices. The simulations and implementations serve as a validation of the theoretical results and provide insights for further research directions.

In the following sections, we summarize the topics discussed in this dissertation and briefly outline its contributions. We conclude with open problems and future research directions.

7.1 Summary of the Dissertation

7.1.1 Theoretical study

The main goal of this thesis is to introduce new codes for private coded computing with straggler mitigation. We notice that communication efficient secret sharing schemes are a natural candidate for such codes. Therefore, we first construct communication efficient secret sharing codes and analyze their performance in private coded computing.

Communication efficient secret sharing In communication efficient secret sharing (CE-SS), a master shares a secret with n parties. A legitimate user contacts d parties, such that $k \leq d \leq n$ for some parameter k, to decode the secret. The goal is to minimize the read and download overheads for the user. We introduce a new class of linear CE-SS codes called *Staircase Codes*. We divide Staircase codes into three classes. The most general construction is called Δ-universal Staircase code, it allows the user to achieve optimal costs universally for all values of d in a given set $\Delta \subseteq \{k, \dots, n\}$. Two other classes of interest are universal Staircase codes that allow the user to achieve optimal costs for all values of $d \in \{k, \dots, n\}$; and Staircase codes for fixed d that allow the user to achieve optimal costs for a predetermined value of d. All Staircase code constructions can store a secret of maximal size and require a small finite field $\mathbb{F}_q$ of size $q > n$. However, these code constructions require dividing the shares into α symbols. The difference between the three constructions is the value of α. The value of α increases with the cardinality of the set Δ.

Private coded computing via Staircase codes We analyze the performance of Staircase codes in a private coded computing setting where the workers have a fixed resources available. In many systems of interest, communication is the main bottleneck. Therefore, minimizing the communication cost by using Staircase codes significantly reduces the delays caused by stragglers. We show that universal Staircase codes outperform any codes that set a threshold on the number of stragglers. More precisely, Staircase codes allow flexibility in the number of stragglers up to a given threshold. This flexibility is accompanied by a universal reduction of the communication cost at the master for any given number of stragglers. The reduction in communication results from saving resources spent on computing the random keys used only for privacy purposes. We assume an *i.i.d.* shifted exponential distribution of the service time of the workers. We derive an upper bound and a lower bound on the master's mean waiting time when using Staircase codes. We characterize the distribution of the master's waiting time, and its mean, for systems with $n = k - 1$ and $n = k - 2$. Moreover, we derive an expression that can give the exact distribution, and the mean, of the waiting time of the master for general (n, k, z) systems.

Adaptive private coded computing We consider the setting in which the resources at the workers are time-varying. Applications include clusters with variability in the computation workload and network congestion at the workers. Other applications include Internet of Things networks and Edge computing where the workers also enjoy great variability due to the difference in the nature of the devices (phone, tablet, sensor, etc). The goal here is to develop a private coded computation scheme that adaptively assigns the sub-tasks to the workers. We aim to assign a task (number of sub-tasks) proportional to the estimated resources at the worker. We design a private and rateless adaptive coded computation (PRAC) scheme that adaptively assigns sub-tasks to the workers to efficiently use their available resources while maintaining the privacy of the data. PRAC is based on the use of Fountain codes coupled with an MDS code to ensure privacy. We show that under this setting, PRAC outperforms all private coded computing schemes that pre-allocate the tasks to the workers.

Stochastic gradient coding We study the effect of coding on the convergence of non-private coded computing schemes. In this setting, the master replicates the data to the workers without further preprocessing. The workers can thus compute any function of the data they possess. We focus on computing additively separable loss functions which encompass many loss functions used in the machine learning literature. We propose stochastic gradient coding, a new strategy that replicates the data to the workers in an unequal manner. Each worker computes a linear combination of the gradient of the loss function on the data it has received and sends the result to the master. The master sums the results received from the non-stragglers to compute an estimate of the true gradient. We show that with a constant amount of redundancy SGC maintains a good convergence rate and mirrors known results from the literature on stochastic gradient descent. Also, we show that SGC outperforms other approximate gradient coding schemes in the regime where lots of workers are stragglers.

Private information retrieval We study robust private information retrieval (PIR). In this setting, the data is public and owned by the workers. The master wants to retrieve a file by querying n workers without revealing the identity of the retrieved file to the workers. We assume the workers can collude and consider the setting in which the workers might be stragglers, i.e., slow or unresponsive. The main cost of a PIR scheme is the download cost incurred by the master to retrieve the file. PIR capacity, i.e., minimum download cost, depends on the number of stragglers. We bridge communication efficient secret sharing to the robust private information retrieval problem. We show that a communication efficient secret sharing scheme achieves PIR capacity simultaneously for any number of stragglers. As a result, we introduce Staircase-PIR, a family of robust PIR codes that achieves capacity simultaneously for any number of stragglers, up to a given threshold.

7.1.2 Simulations and implementations

We complement the theoretical study presented in this dissertation with extensive simulations and implementations on MATLAB, Amazon EC2 clusters, and Android devices. We simulate Staircase codes, PRAC, and SGC on MATLAB to supplement our theoretical findings. Furthermore, we implement Staircase codes on Amazon EC2 clusters. We observe that Staircase codes provide the most latency reduction when the workers and the master are geographically far from each other. Also, we notice that our assumption of equally dividing the response time between subtasks does not hold for large tasks. We implement PRAC on Android devices to emulate a heterogeneous environment. We observe that PRAC outperforms Staircase codes, and thus all threshold secret sharing schemes when the workers have different computation and communication power. However, when the workers have similar response time (more homogeneous) Staircase codes outperform PRAC as expected. This is due to the overhead introduced by using Fountain codes in PRAC.

7.2 Future Directions

Bounds on the share size We pointed out that Staircase codes require dividing the secret (and the shares) into α symbols. So far, no bounds on the values of α are known. However, we conjecture that the value of α required by Staircase codes is optimal if one wishes to reduce the read overhead. We leave the problem of finding bounds on α and constructing CE-SS codes with possibly smaller values of α as an open problem.

Minimizing latency in homogeneous private coded computing In terms of latency in private coded computing, Staircase codes reduce the master's waiting time by minimizing the download cost. However, they are not designed to minimize latency. The problem of designing codes that minimize the latency remains open in general. Further, equally dividing the response time of the workers among the subtasks seems to not reflect our observations from Amazon EC2 implementations. We expect that modeling the response time of the subtasks as shifted exponential random variables with smaller shifts and rates would be a better representation of our observations.

Malicious adversaries Throughout the dissertation, we assumed that the colluding workers only intend to eavesdrop on the master's data. However, the workers could be malicious and corrupt the results sent to the master. Security against malicious workers in coded computing is crucial and its literature is growing. Staircase codes can be readily extended to this setting due to their error correction capability inherent from their similarity to Reed-Solomon codes. Error correction assumes that the adversary has full knowledge of the data being stored and computed. In distributed storage, this assumption implies a penalty on the size of the stored secret that is twice the number of compromised workers. The more interesting case to study is when the adversary's knowledge is limited to the information obtained by the compromised workers. Leveraging the limitation of the adversary's knowledge reduces the penalty on the size of the stored secret. It is not clear how to extend such ideas to distributed computing and PIR.

Private coded computation for machine learning applications In this dissertation, we restricted our focus to private linear coded computing which is used in many machine learning algorithms. However, constructing private coded computing schemes for general gradient-descent algorithms with flexible straggler tolerance remains open. We note that progress is being made in this direction. In [103,148] the authors introduce private and secure coded computing schemes for multivariate polynomials. Yet those codes still impose a threshold on the number of stragglers and need not extend to any gradient-descent like algorithm. We are actively investigating this problem.

Another problem of interest is the problem of hiding the attribute vector $\mathbf{x}$. Here, we provide a "systems" solution to hide the attribute vector in Appendix A by replicating the computations on a disjoint set of workers. A more efficient way is to encode both A and $\mathbf{x}$ using secret sharing schemes. This problem is referred to as private matrix-matrix multiplication and is currently an active research topic, see [19] and references within. Despite the improvement on this topic, all introduced codes set a threshold on the number of stragglers, which is restrictive as it forces the workers to spend resources on computing the randomness used only for privacy.

References

[1] E. Learned-Miller, G. B. Huang, A. Roy Chowdhury, H. Li, and G. Hua, "Labeled faces in the wild: A survey," in *Advances in face detection and facial image analysis.* Springer, 2016, pp. 189–248.

[2] "How much data is generated each day," 2019. [Online]. Available: "https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/"

[3] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 119–129.

[4] R. Cramer, I. B. Damgrd, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, 1st ed. New York, NY, USA: Cambridge University Press, 2015.

[5] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[6] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[7] S. Dutta, V. Cadambe, and P. Grover, "Short-Dot: Computing large linear transforms distributedly using coded short dot products," in *Advances In Neural Information Processing Systems (NIPS)*, 2016, pp. 2100–2108.

[8] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[9] G. R. Blakley, "Safeguarding cryptographic keys," in *Proceedings of the National Computer Conference*, vol. 48, 1979, pp. 313–317.

[10] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learning (ICML)*, 2017, pp. 3368–3376.

[11] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed-Solomon codes," in *IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 2027–2031.

[12] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *International Conference on Machine Learning (ICML)*, vol. 12, 2018, p. 9716p.

[13] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic mds codes and expander graphs," *arXiv preprint arXiv:1707.03858*, 2017.

[14] Z. Charles, D. Papailiopoulos, and J. Ellenberg, "Approximate gradient coding via sparse random graphs," *arXiv preprint arXiv:1711.06771*, 2017.

[15] R. K. Maity, A. S. Rawat, and A. Mazumdar, "Robust gradient descent via moment encoding with LDPC codes," *arXiv preprint arXiv:1805.08327*, 2018.

[16] S. Wang, J. Liu, and N. Shroff, "Fundamental limits of approximate gradient coding," *arXiv preprint arXiv:1901.08166*, 2019.

[17] H. Wang, Z. Charles, and D. Papailiopoulos, "ErasureHead: Distributed gradient descent without delays using approximate gradient coding," *arXiv preprint arXiv:1901.09671*, 2019.

[18] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2010, pp. 48–59.

[19] R. G. D'Oliveira, S. E. Rouayheb, and D. Karpuk, "Gasp codes for secure distributed matrix multiplication," *arXiv preprint arXiv:1812.09962*, 2018.

[20] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[21] J. Kakar, S. Ebadifar, and A. Sezgin, "Rate-efficiency and straggler-robustness through partition in distributed two-sided secure matrix computation," *arXiv preprint arXiv:1810.13006*, 2018.

[22] H. Wang and D. S. Wong, "On secret reconstruction in secret sharing schemes," *IEEE Transactions on Information Theory*, vol. 54, no. 1, pp. 473–480, Jan 2008.

[23] W. Huang, M. Langberg, J. Kliewer, and J. Bruck, "Communication efficient secret sharing," *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 7195–7206, 2016.

[24] R. Bitar and S. E. Rouayheb, "Staircase codes for secret sharing with optimal communication and read overheads," in *IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 1396–1400.

[25] R. Bitar and S. El Rouayheb, "Staircase codes for secret sharing with optimal communication and read overheads," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 933–943, 2018.

[26] G. Liang and U. C. Kozat, "TOFEC: Achieving optimal throughput-delay trade-off of cloud storage using erasure codes," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2014, pp. 826–834.

[27] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing latency for secure distributed computing," in *IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2900–2904.

[28] ——, "Minimizing latency for secure coded computing using secret sharing via staircase codes," *arXiv preprint arXiv:1802.02640*, 2018.

[29] M. Luby, "Lt codes," in *null*. IEEE, 2002, p. 271.

[30] A. Shokrollahi, "Raptor codes," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2551–2567, 2006.

[31] D. J. MacKay, "Fountain codes," *IEE Proceedings-Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.

[32] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, 2019.

[33] Y. Keshtkarjahromi, Y. Xing, and H. Seferoglu, "Dynamic heterogeneity-aware coded cooperative computation at the edge," in *26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 23–33.

[34] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. El Rouayheb, and H. Seferoglu, "PRAC: Private and Rateless Adaptive Coded Computation at the Edge," in *SPIE - Defense + Commercial Sensing*, vol. 11013, 2019. [Online]. Available: https://doi.org/10.1117/12.2519768

[35] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. E. Rouayheb, and H. Seferoglu, "Private and rateless adaptive coded matrix-vector multiplication," *arXiv preprint arXiv:1909.12611*, 2019.

[36] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," *arXiv preprint arXiv:1109.5647*, 2011.

[37] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[38] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[39] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," *arXiv preprint arXiv:1604.00981*, 2016.

[40] S. Horii, T. Yoshida, M. Kobayashi, and T. Matsushima, "Distributed stochastic gradient descent using LDGM codes," *arXiv preprint arXiv:1901.04668*, 2019.

[41] S. Kadhe, O. O. Koyluoglu, and K. Ramchandran, "Gradient coding based on block designs for mitigating adversarial stragglers," *arXiv preprint arXiv:1904.13373*, 2019.

[42] H. Sun and S. A. Jafar, "The capacity of private computation," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3880–3897, 2018.

[43] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *IEEE Symposium on Foundations of Computer Science*, 1995, pp. 41–50.

[44] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 965–981, 1998.

[45] A. Beimel and Y. Stahl, "Robust information-theoretic private information retrieval," in *International Conference on Security in Communication Networks.* Springer, 2002, pp. 326–341.

[46] H. Sun and S. A. Jafar, "The capacity of robust private information retrieval with colluding databases," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2361–2370, April 2018.

[47] ——, "The capacity of robust private information retrieval with colluding databases," *arXiv preprint arXiv:1605.00635*, 2016.

[48] C. Devet, I. Goldberg, and N. Heninger, "Optimally robust private information retrieval." in *USENIX Security Symposium*, 2012, pp. 269–283.

[49] R. Tajeddine and S. El Rouayheb, "Robust private information retrieval on coded data," *arXiv preprint arXiv:1707.09916v1*, 2017.

[50] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, and C. Hollanti, "Robust private information retrieval from coded systems with byzantine and colluding servers," *arXiv preprint arXiv:1802.03731*, 2018.

[51] K. Banawan and S. Ulukus, "The capacity of private information retrieval from coded databases," *arXiv preprint arXiv:1609.08138*, 2016.

[52] ——, "The capacity of private information retrieval from byzantine and colluding databases," *arXiv preprint arXiv:1706.01442*, 2017.

[53] R. J. McEliece and D. V. Sarwate, "On sharing secrets and reed-solomon codes," *Communications of the ACM*, vol. 24, no. 9, pp. 583–584, 1981.

[54] H.-Y. Chien, J. Jinn-Ke, and Y.-M. Tseng, "A practical (t, n) multi-secret sharing scheme," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 83, no. 12, pp. 2762–2765, 2000.

[55] E. D. Karnin, J. W. Greene, and M. E. Hellman, "On secret sharing systems," *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 35–41, 1983.

[56] C.-P. Lai and C. Ding, "Several generalizations of Shamir's secret sharing scheme," *International Journal of Foundations of Computer Science*, vol. 15, no. 02, pp. 445–458, 2004.

[57] C.-C. Yang, T.-Y. Chang, and M.-S. Hwang, "A (t, n) multi-secret sharing scheme," *Applied Mathematics and Computation*, vol. 151, no. 2, pp. 483–490, 2004.

[58] J. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," in *Proceedings on Advances in Cryptology.* Springer-Verlag New York, Inc., 1990, pp. 27–35.

[59] M. Ito, A. Saito, and T. Nishizeki, "Secret sharing scheme realizing general access structure," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 72, no. 9, pp. 56–64, 1989.

[60] E. F. Brickell, "Some ideal secret sharing schemes," in *Advances in Cryptology–EUROCRYPT'89*, 1990, pp. 468–475.

[61] C. Padró, "Lecture notes in secret sharing." *IACR Cryptology ePrint Archive*, vol. 2012, p. 674, 2012.

[62] A. Beimel, "Secret-sharing schemes: a survey," in *Coding and Cryptology.* Springer, 2011, pp. 11–46.

[63] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing.* Cambridge, England: Cambridge University Press, 2015.

[64] Z. Zhang, Y. M. Chee, S. Ling, M. Liu, and H. Wang, "Threshold changeable secret sharing schemes revisited," *Theoretical Computer Science*, vol. 418, pp. 106–115, 2012.

[65] W. Huang, M. Langberg, J. Kliewer, and J. Bruck, "Communication efficient secret sharing," *arXiv preprint arXiv:1505.07515*, May 2015.

[66] R. Bitar and S. E. Rouayheb, "Staircase codes for secret sharing with optimal communication and read overheads," *arXiv preprint arXiv:1512.02990*, Dec 2015.

[67] W. Huang, M. Langberg, J. Kliewer, and J. Bruck, "Communication efficient secret sharing," *arXiv preprint arXiv:1505.07515v2*, April 2016.

[68] N. B. Shah, K. V. Rashmi, and K. Ramchandran, "Distributed secret dissemination across a network," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 7, pp. 1206–1216, Oct 2015.

[69] A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, "Centralized repair of multiple node failures with applications to communication efficient secret sharing," *arXiv preprint arXiv:1603.04822*, 2016.

[70] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5227–5239, 2011.

[71] N. B. Shah, K. Rashmi, and P. V. Kumar, "Information-theoretically secure regenerating codes for distributed storage," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2011, pp. 1–5.

[72] S. E. Rouayheb, E. Soljanin, and A. Sprintson, "Secure network coding for wiretap networks of type II," *IEEE Transactions on Information Theory*, vol. 58, no. 3, pp. 1361–1371, 2012.

[73] Https://setiathome.berkeley.edu.

[74] Https://foldingathome.stanford.edu.

[75] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[76] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[77] A. Rényi, "On the theory of order statistics," *Acta Mathematica Academiae Scientiarum Hungarica*, vol. 4, no. 3-4, pp. 191–231, 1953.

[78] Https://aws.amazon.com/ec2.

[79] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments." in *Osdi*, vol. 8, 2008, p. 7.

[80] G. Ananthanarayanan, S. Kandula, A. G. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, "Reining in the outliers in map-reduce clusters using mantri." in *OSDI*, vol. 10, 2010, p. 24.

[81] S. Narayanamurthy, M. Weimer, D. Mahajan, T. Condie, S. Sellamanickam, and S. S. Keerthi, "Towards resource-elastic machine learning," in *NIPS 2013 BigLearn Workshop*, 2013.

[82] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2011, pp. 693–701.

[83] I. Mitliagkas, C. Zhang, S. Hadjis, and C. Ré, "Asynchrony begets momentum, with an application to deep learning," in *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on.* IEEE, 2016, pp. 997–1004.

[84] B. Recht, C. Re, J. Tropp, and V. Bittorf, "Factoring nonnegative matrices with linear programs," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1214–1222.

[85] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin, "A fast parallel SGD for matrix factorization in shared memory systems," in *7th ACM conference on Recommender systems*, 2013, pp. 249–256.

[86] H. Yun, H.-F. Yu, C.-J. Hsieh, S. Vishwanathan, and I. Dhillon, "NOMAD: Non-locking, stOchastic Multi-machine algorithm for Asynchronous and Decentralized matrix completion," *VLDB Endowment*, vol. 7, no. 11, pp. 975–986, 2014.

[87] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm." *Journal of Machine Learning Research*, vol. 16, no. 285-322, pp. 1–5, 2015.

[88] J. Duchi, M. I. Jordan, and B. McMahan, “Estimation, optimization, and parallelism when data is sparse,” in *Advances in Neural Information Processing Systems*, 2013, pp. 2832–2840.

[89] Y.-x. Wang, V. Sadhanala, W. Dai, W. Neiswanger, S. Sra, and E. P. Xing, “Asynchronous parallel block-coordinate frank-wolfe,” *stat*, vol. 1050, p. 22, 2014.

[90] C.-J. Hsieh, H.-F. Yu, and I. S. Dhillon, “Passcode: Parallel asynchronous stochastic dual co-ordinate descent.” in *International Conference on Machine Learning (ICML)*, vol. 15, 2015, pp. 2370–2379.

[91] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan, “Perturbed iterate analysis for asynchronous stochastic optimization,” *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2202–2229, 2017.

[92] T. M. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, “Project Adam: Building an efficient and scalable deep learning training system.” in *OSDI*, vol. 14, 2014, pp. 571–582.

[93] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, “Large scale distributed deep networks,” in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[94] L. Huang, S. Pawar, H. Zhang, and K. Ramchandran, “Codes can reduce queueing delay in data centers,” in *IEEE International Symposium on Information Theory (ISIT)*, 2012.

[95] G. Joshi, Y. Liu, and E. Soljanin, “Coding for fast content download,” in *50th Annual Allerton Conference on Communication, Control, and Computing*, 2012.

[96] S. Kadhe, E. Soljanin, and A. Sprintson, “Analyzing the download time of availability codes,” in *IEEE International Symposium on Information Theory (ISIT)*, 2015.

[97] Q. Yu, M. Maddah-Ali, and S. Avestimehr, “Polynomial codes: an optimal design for high-dimensional coded matrix multiplication,” in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 4403–4413.

[98] S. Dutta, V. Cadambe, and P. Grover, “Coded convolution for parallel and distributed computing within a deadline,” in *IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2403–2407.

[99] Y. Yang, P. Grover, and S. Kar, “Computing linear transformations with unreliable components,” *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3729–3756, 2017.

[100] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, “A unified coding framework for distributed computing with straggling servers,” in *IEEE Globecom Workshops (GC Wkshps)*, 2016, pp. 1–6.

[101] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, “A fundamental tradeoff between computation and communication in distributed computing,” *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2018.

[102] H. Yang and J. Lee, “Secure distributed computing with straggling servers using polynomial codes,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 141–150, 2019.

[103] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, “Lagrange coded computing: Optimal design for resiliency, security, and privacy,” in *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019, pp. 1215–1225.

[104] J. So, B. Guler, A. S. Avestimehr, and P. Mohassel, "Codedprivateml: A fast and privacy-preserving framework for distributed machine learning," *arXiv preprint arXiv:1902.00641*, 2019.

[105] Q. Yu and A. S. Avestimehr, "Harmonic coding: An optimal linear code for privacy-preserving gradient-type computation," *arXiv preprint arXiv:1904.13206*, 2019.

[106] H. Takabi, E. Hesamifard, and M. Ghasemi, "Privacy preserving multi-party machine learning with homomorphic encryption," in 29*th Annual Conference on Neural Information Processing Systems (NIPS)*, 2016.

[107] R. Hall, S. E. Fienberg, and Y. Nardi, "Secure multiple linear regression based on homomorphic encryption," *Journal of Official Statistics*, vol. 27, no. 4, p. 669, 2011.

[108] L. Kamm, D. Bogdanov, S. Laur, and J. Vilo, "A new way to protect privacy in large-scale genome-wide association studies," *Bioinformatics*, vol. 29, no. 7, pp. 886–893, 2013.

[109] S. Gade and N. H. Vaidya, "Private learning on networks: Part ii," *arXiv preprint arXiv:1703.09185*, 2017.

[110] Y. Keshtkarjahromi, Y. Xing, and H. Seferoglu, "Dynamic heterogeneity-aware coded cooperative computation at the edge," in *26th International Conference on Network Protocols (ICNP)*, 2018.

[111] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler mitigation in distributed optimization through data encoding," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5434–5442.

[112] M. F. Aktas, P. Peng, and E. Soljanin, "Effective straggler mitigation: Which clones should attack and when?" *ACM SIGMETRICS Performance Evaluation Review*, vol. 45, no. 2, pp. 12–14, 2017.

[113] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586–2595, 2017.

[114] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[115] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[116] S. Lin and D. Costello, *Error-Correcting Codes.* Prentice-Hall, Inc, 1983.

[117] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes.* Elsevier, 1977.

[118] G. A. Seber and A. J. Lee, *Linear regression analysis.* John Wiley & Sons, 2012, vol. 329.

[119] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[120] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo, "Reed-solomon forward error correction (FEC) schemes," Tech. Rep., 2009.

[121] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[122] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84 – 106, 2013.

[123] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 369–392, 2014.

[124] Y. Geng, W. Hu, Y. Yang, W. Gao, and G. Cao, "Energy-efficient computation offloading in cellular networks," in *IEEE International Conference on Network Protocols (ICNP)*, 2015.

[125] R. K. Lomotey and R. Deters, "Architectural designs from mobile cloud computing to ubiquitous cloud computing - survey," in *IEEE World Congress on Services*, 2014.

[126] T. Penner, A. Johnson, B. V. Slyke, M. Guirguis, and Q. Gu, "Transient clouds: Assignment and collaborative execution of tasks on mobile devices," in *IEEE Global Communications Conference*, 2014.

[127] P. Peng and E. Soljanin, "On distributed storage allocations of large files for maximum service rate," in *56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2018, pp. 784–791.

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

[129] L. Chen, Z. Charles, D. Papailiopoulos *et al.*, "DRACO: Robust distributed training via redundant gradients," *arXiv preprint arXiv:1803.09877*, 2018.

[130] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1988–1992.

[131] E. Ozfaturay, D. Gunduz, and S. Ulukus, "Speeding up distributed gradient descent by utilizing non-persistent stragglers," *arXiv preprint arXiv:1808.02240*, 2018.

[132] N. Ferdinand and S. Draper, "Anytime stochastic gradient descent: A time to hear from all the workers," in *56th Annual Allerton Conference on Communication, Control, and Computing*, 2018, pp. 552–559.

[133] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2018, pp. 857–866.

[134] U. Sheth, S. Dutta, M. Chaudhari, H. Jeong, Y. Yang, J. Kohonen, T. Roos, and P. Grover, "An application of storage-optimal MatDot codes for coded matrix multiplication: Fast k-nearest neighbors estimation," in *IEEE International Conference on Big Data (Big Data)*, 2018, pp. 1113–1120.

[135] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *arXiv preprint, arXiv:1806.00939*, 2018.

[136] Y. Keshtkarjahromi, R. Bitar, V. Dasari, S. E. Rouayheb, and H. Seferoglu, "Secure coded cooperative computation at the heterogeneous edge against byzantine attacks," in *IEEE Global Communication Conference (GLOBECOM)*, 2019.

[137] G. Ananthanarayanan, S. Kandula, A. G. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, "Reining in the outliers in map-reduce clusters using mantri." in *OSDI*, vol. 10, no. 1, 2010, p. 24.

[138] L. Bottou, “Online learning and stochastic approximations,” *On-line learning in neural networks*, vol. 17, no. 9, p. 142, 1998.

[139] S. Shalev-Shwartz and A. Tewari, “Stochastic methods for ℓ_1-regularized loss minimization,” *Journal of Machine Learning Research*, vol. 12, no. Jun, pp. 1865–1892, 2011.

[140] K. Gimpel, D. Das, and N. A. Smith, “Distributed asynchronous online learning for natural language processing,” in *40th Conference on Computational Natural Language Learning*, 2010, pp. 213–222.

[141] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, “PEGASOS: Primal estimated sub-gradient solver for SVM,” *Mathematical programming*, vol. 127, no. 1, pp. 3–30, 2011.

[142] M. Amiri and D. Gündüz, “Computation scheduling for distributed machine learning with straggling workers,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8177–8181.

[143] S. El Rouayheb and K. Ramchandran, “Fractional repetition codes for repair in distributed storage systems,” in *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2010, pp. 1510–1517.

[144] T. Strohmer and R. Vershynin, “A randomized Kaczmarz algorithm with exponential convergence,” *Journal of Fourier Analysis and Applications*, vol. 15, no. 2, p. 262, 2009.

[145] D. Needell, R. Ward, and N. Srebro, “Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm,” in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 1017–1025.

[146] D. Needell and R. Ward, “Batched stochastic gradient descent with weighted sampling,” in *International Conference Approximation Theory*, 2016, pp. 279–306.

[147] “SGC GitHub repository,” https://github.com/RawadB01/SGC.

[148] H. Yang and J. Lee, “Secure distributed computing with straggling servers using polynomial codes,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 141–150, 2019.

[149] S. Li, S. M. M. Kalan, Q. Yu, M. Soltanolkotabi, and A. S. Avestimehr, “Polynomially coded regression: Optimal straggler mitigation via data encoding,” *arXiv preprint arXiv:1805.09934*, 2018.

[150] E. Ozfatura, S. Ulukus, and D. Gündüz, “Distributed gradient descent with coded partial gradient computations,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3492–3496.

[151] S. Wang, J. Liu, and N. Shroff, “Coded sparse matrix multiplication,” *arXiv preprint arXiv:1802.03430*, 2018.

[152] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, “Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes,” in *IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1993–1997.

[153] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover, “On the optimal recovery threshold of coded matrix multiplication,” in *55th Annual Allerton Conference on Communication, Control, and Computing*, 2017, pp. 1264–1270.

[154] A. Mallick, M. Chaudhari, and G. Joshi, “Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication,” *arXiv preprint arXiv:1804.10331*, 2018.

[155] S. Wang, J. Liu, N. Shroff, and P. Yang, "Fundamental limits of coded linear transform," *arXiv preprint arXiv:1804.09791*, 2018.

[156] M. F. Aktas, P. Peng, and E. Soljanin, "Effective straggler mitigation: Which clones should attack and when?" *ACM SIGMETRICS Performance Evaluation Review*, vol. 45, no. 2, pp. 12–14, 2017.

[157] D. Wang, G. Joshi, and G. Wornell, "Using straggler replication to reduce latency in large-scale parallel computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 3, pp. 7–11, 2015.

[158] K. Lee, N. B. Shah, L. Huang, and K. Ramchandran, "The MDS queue: Analysing the latency performance of erasure codes," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 2822–2842, 2017.

[159] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[160] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, "Better mini-batch algorithms via accelerated gradient methods," in *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 1647–1655.

[161] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 873–881.

[162] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 165–202, 2012.

[163] O. Shamir and N. Srebro, "Distributed stochastic optimization and learning," in *52nd Annual Allerton Conference on Communication, Control, and Computing*, 2014, pp. 850–857.

[164] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods.* Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[165] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD," *arXiv preprint arXiv:1803.01113*, 2018.

[166] S. Chaturapruek, J. C. Duchi, and C. Ré, "Asynchronous stochastic convex optimization: the noise is in the noise and SGD don't care," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1531–1539.

[167] D. Woodruff and S. Yekhanin, "A geometric approach to information-theoretic private information retrieval," in *Computational Complexity, 2005. Proceedings. Twentieth Annual IEEE Conference on.* IEEE, 2005, pp. 275–284.

[168] W. Gasarch, "A survey on private information retrieval," in *Bulletin of the EATCS.* Citeseer, 2004.

[169] S. Yekhanin, "Private information retrieval," *Communications of the ACM*, vol. 53, no. 4, pp. 68–73, 2010.

[170] A. Beimel, Y. Ishai, . E. Kushilevitz, and J.-F. Raymond, "Breaking the $o(n^{1/(2k-1)})$ barrier for information-theoretic retrieval," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* IEEE, 2002, pp. 261–270.

[171] N. Shah, K. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," in *2014 IEEE International Symposium on Information Theory.* IEEE, 2014, pp. 856–860.

[172] T. Chan, S.-W. Ho, and H. Yamamoto, "Private information retrieval for coded storage," in *2015 IEEE International Symposium on Information Theory (ISIT).* IEEE, June 2015, pp. 2842–2846.

[173] R. Tajeddine and S. El Rouayheb, "Private information retrieval from mds coded data in distributed storage systems," in *IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 1411–1415.

[174] R. Freij-Hollanti, O. Gnilke, C. Hollanti, and D. Karpuk, "Private information retrieval from coded databases with colluding servers," *arXiv preprint arXiv:1611.02062*, 2016.

[175] S. Kumar, E. Rosnes, and A. G. i Amat, "Private information retrieval in distributed storage systems using an arbitrary linear code," in *Information Theory (ISIT), 2017 IEEE International Symposium on.* IEEE, 2017, pp. 1421–1425.

[176] R. G. D'Oliveira and S. El Rouayheb, "Lifting private information retrieval from two to any number of messages," *arXiv preprint arXiv:1802.06443*, 2018.

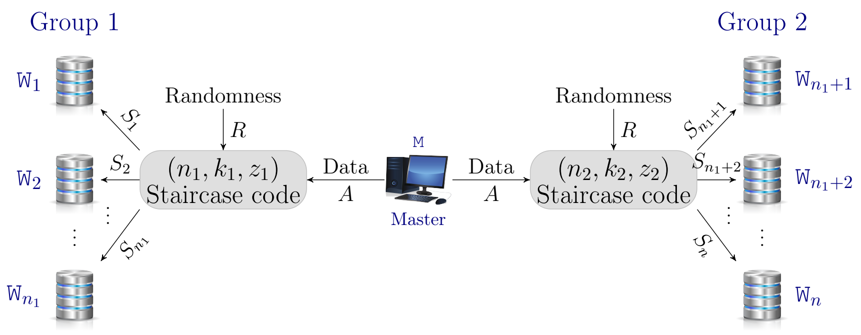
[177] A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, "Centralized repair of multiple node failures with applications to communication efficient secret sharing," *IEEE Transactions on Information Theory*, vol. 64, no. 12, pp. 7529–7550, 2018.

[178] Y. Keshtkarjahromi, Y. Xing, and H. Seferoglu, "Dynamic heterogeneity-aware coded cooperative computation at the edge," *arXiv preprint, rXiv:1801.04357v3*, 2018.

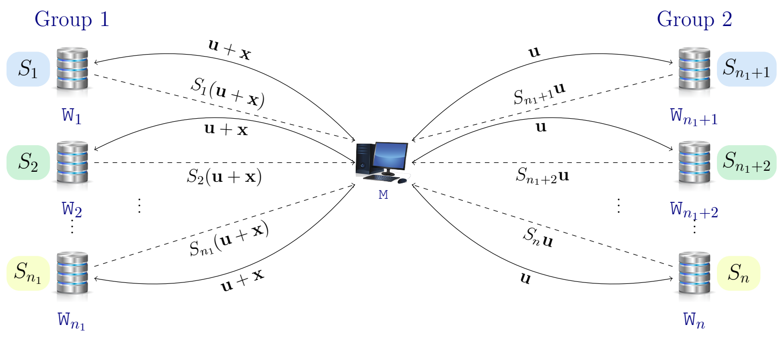
Appendices

Appendix A

Hiding the attribute vectors



(a) M encodes A using an (n_1, k_1, z_1) and (n_2, k_2, z_2) Staircase codes and distribute the obtained shares to the workers.



(b) M sends $\mathbf{x}+\mathbf{u}$ and $\mathbf{u}$ to the workers of group 1 and 2, respectively. $\mathtt{W}_i$ computes $S_i\mathbf{x}$ and sends the result to M.

Figure A.1: Secure distributed matrix-vector multiplication with n workers, where the master needs to hide both the data A and the matrix $\mathbf{x}$. M divides the workers intro two disjoint groups of cardinality n_1 and n_2, respectively, such that $n_1 + n_2 = n$. Now M deals with the groups as two separate (n_1, k_1, z_1) and (n_2, k_2, z_2) systems, where $z_1 < k_1 < n_1$ and $z_2 < k_2 < n_2$. To hide $\mathbf{x}$, M generates a random vector $\mathbf{u}$ and sends $\mathbf{x}+\mathbf{u}$ to group 1 and $\mathbf{u}$ to group 2. Hence, M decodes $A\mathbf{x}$ after decoding $A\mathbf{u}$ and $A(\mathbf{x}+\mathbf{u})$.

Throughout Chapter 3 and Chapter 4 we assumed privacy over one iteration of the computation, i.e., the Master needs to hide only the data matrix A. In the following we describe how our schemes can be generalized to achieve privacy over the whole algorithm, i.e., the Master needs to hide A and the attribute vectors $\mathbf{x}^1, \mathbf{x}^2, \dots$. Since the algorithms are iterative, we focus on one iteration and the remaining follows in a similar manner. Let A be an $m \times \ell$ matrix and $\mathbf{x}$ be an $\ell \times 1$ vector that the Master $\mathtt{M}$ wishes to distributively multiply. Let n be the number of workers $\mathtt{W}_i$, $i = 1, \dots, n$, that volunteer to help $\mathtt{M}$. The idea is to divide the workers into disjoint groups and use classical linear sharing to send the vector $\mathbf{x}$ to the workers. Therefore, the Master asks each group to securely multiply A by a vector that is statistically independent of $\mathbf{x}$. Then, the Master decodes $A\mathbf{x}$ from the results of the received multiplications. For simplicity, we explain this idea through an example. Assume the master wants to divide the workers into only two groups. $\mathtt{M}$ divides the workers into two groups of cardinality n_1 and n_2 such that $n_1 + n_2 = n$. Afterwards, $\mathtt{M}$ chooses $z_1 < k_1 < n_1$ and $z_2 < k_2 < n_2$ and encodes A into n shares using an (n_1, k_1, z_1) and an (n_2, k_2, z_2) Staircase codes (or classical secret sharing codes). Thereafter, $\mathtt{M}$ distributes the shares to the workers such that the groups form two disjoint (n_1, k_1, z_1) and (n_2, k_2, z_2) systems. To hide $\mathbf{x}$, $\mathtt{M}$ generates a random vector $\mathbf{u}$ of same size as $\mathbf{x}$ and sends $\mathbf{x} + \mathbf{u}$ to the first group and $\mathbf{u}$ to the second group. According to our scheme, $\mathtt{M}$ decodes $A(\mathbf{x} + \mathbf{u})$ and $A\mathbf{u}$ after receiving enough responses from the workers of each group. Hence, $\mathtt{M}$ can decode $A\mathbf{x}$. Note that no information about $\mathbf{x}$ is revealed because it is one-time padded by $\mathbf{u}$. We illustrate the idea in Figure A.1. In the general case, $\mathtt{M}$ divides the workers into g groups of cardinality $n_1, n_2, \dots n_g$ such that $n_1 + n_2 + \cdots + n_g = n$. The master decides on how many groups can collude z_g and how many straggling groups s_g it wants to tolerate and therefore encodes $\mathbf{x}$ into a $(g, g - s_g, z_g)$ linear classical secret sharing and sends each share to a different group. The remaining follows as explained above for both schemes explained in Chapter 3 and Chapter 4.

Appendix B

Additional Proofs for Chapter 3

B.1 Proof of Theorem 3.2

For the clarity of presentation, we restate Theorem 3.2.

Theorem (Exact expression of $\mathbb{E}[T_{\mathrm{SC}}]$ for systems with up to 2 stragglers)**.** *The mean waiting time of the Master for $(k+1,k,z)$ and $(k+2,k,z)$ systems is given in (B.1) and (B.2), respectively.*

$$\mathbb{E}\left[T_{SC}(k+1,k,z)\right] = \frac{c}{k-z+1} + \frac{1}{\lambda}\sum_{i=1}^{k+1}(-1)^i\binom{k+1}{i}\left[\frac{i\exp\left(\frac{-\lambda c}{k-z}\right)}{(k-z)i+1} - \frac{1}{(k-z+1)i}\right]. \tag{B.1}$$

$$\begin{aligned}\mathbb{E}[T_{SC}(k+2,k,z)] = {} & \mathbb{E}[T_{SC}(k+2,k+1,z)] \\ & + \frac{1}{\lambda}\sum_{i=2}^{k+2}(-1)^i\binom{k+2}{i}\binom{i}{2}\left[\frac{\exp\left(-\frac{4\lambda c}{k-z}\right)}{(k-z)i+4} - \frac{2\exp\left(-\frac{3\lambda c}{k-z}\right)}{(k-z)i+3}\right].\end{aligned} \tag{B.2}$$

We derive the expression of the Master's mean waiting time for $(n,k,z) = (k+1,k,z)$ and $(n,k,z) = (k+2,k,z)$ systems. Applying Theorem 3.4 for the case of $n = k+1$, we get

$$\bar{F}_{T_{\mathrm{SC}}(k+1,k,z)}(t) = 1 - F_{T'}(t_{k+1})^{k+1} - F_{T'}(t_k)^k\,\bar{F}_{T'}(t_{k+1})(k+1), \quad \text{for } t > 0.$$

Recall that t_k and t_{k+1} are defined as $t_k = \max\left\{t - \frac{c}{k-z}, 0\right\}$ and $t_{k+1} = \max\left\{\frac{k+1-z}{k-z}\left(t - \frac{c}{k+1-z}\right), 0\right\}$.

Since $F_{T'}(0) = 0$, we can compute the Master's mean waiting time $\mathbb{E}\left[T(k+1,k,z)\right]$ as

$$\begin{aligned}\mathbb{E}\left[T_{\mathrm{SC}}(k+1,k,z)\right] &= \int_0^\infty (1-(1-\bar{F}_{T'}(t_{k+1}))^{k+1})dt - \int_0^\infty (1-\bar{F}_{T'}(t_k))^k \bar{F}_{T'}\left(t_{k+1}\right)(k+1)dt,\\ &= \frac{c}{k+1-z} + \int_{\frac{c}{k+1-z}}^\infty (1-(1-\bar{F}_{T'}(t_{k+1}))^{k+1})dt\\ &\quad - \int_{\frac{c}{k-z}}^\infty (1-\bar{F}_{T'}(t_k))^k \bar{F}_{T'}\left(t_{k+1}\right)(k+1)dt.\end{aligned}$$

Using the binomial expansion and integrating the exponential function $\bar{F}_{T'}(t) = \exp(-\lambda(k-z)t)$, we get

$$\mathbb{E}\left[T_{\mathrm{SC}}(k+1,k,z)\right] = \frac{c}{k-z+1} + \frac{1}{\lambda}\sum_{i=1}^{k+1}(-1)^i\binom{k+1}{i}\left[\frac{i\exp\left(\frac{-\lambda c}{k-z}\right)}{(k-z)i+1} - \frac{1}{(k-z+1)i}\right].$$

Similarly, we apply Theorem 3.4 for $n = k+2$ and get

$$\begin{aligned}\bar{F}_{T_{\mathrm{SC}}(k+2,k,z)}(t) &= 1 - F_{T'}(t_{k+2})^{k+2}\\ &\quad - (k+2)\bar{F}_{T'}(t_{k+2})\Big[F_{T'}(t_{k+1})^{k+1} + (k+1)F_{T'}(t_k)^k(\bar{F}_{T'}(t_{k+1}) - \frac{1}{2}\bar{F}_{T'}(t_{k+2}))\Big].\end{aligned}$$

Recall that for $i = k, k+1, k+2$, we define t_i as $t_i \triangleq \max\left\{\frac{i-z}{k-z}\left(t - \frac{c}{i-z}\right)\right\}$. Since $F_{T'}(0) = 0$, we can compute the Master's mean waiting time $\mathbb{E}\left[T_{\mathrm{SC}}(k+2,k,z)\right]$ as

$$\begin{aligned}\mathbb{E}\left[T_{\mathrm{SC}}(k+2,k,z)\right] &= \int_0^\infty (1-F_{T'}(t_{k+2})^{k+2})dt - \int_0^\infty (k+2)\bar{F}_{T'}(t_{k+2})F_{T'}(t_{k+1})^{k+1}dt\\ &\quad - \int_0^\infty (k+2)(k+1)\bar{F}_{T'}(t_{k+2})F_{T'}(t_k)^k\left(\bar{F}_{T'}(t_{k+1}) - \frac{1}{2}\bar{F}_{T'}(t_{k+2})\right)dt\\ &= \frac{c}{k+2-z} + \int_{\frac{c}{k+2-z}}^\infty (1-(1-\bar{F}_{T'}(t_{k+2}))^{k+2})dt\\ &\quad - (k+2)\int_{\frac{c}{k+1-z}}^\infty (1-\bar{F}_{T'}(t_{k+1}))^{k+1}\bar{F}_{T'}\left(t_{k+2}\right)dt\\ &\quad - \binom{k+2}{2}\int_{\frac{c}{k-z}}^\infty (1-\bar{F}_{T'}(t_k))^k\bar{F}_{T'}(t_{k+2})\left(2\bar{F}({}_{T'}t_{k+1}) - \bar{F}_{T'}(t_{k+2})\right)dt.\end{aligned}$$

Using the binomial expansion and integrating the exponential function $\bar{F}_{T'}(t) = \exp(-\lambda(k-z)t)$, we get

$$\mathbb{E}\left[T_{\mathrm{SC}}(k+2,k,z)\right] = \frac{c}{k-z+2} + \sum_{i=1}^{k+2} \frac{(-1)^i \binom{k+2}{i}}{\lambda} \left[\frac{i \exp\left(-\frac{\lambda c}{k-z+1}\right)}{(k-z+1)i+1} - \frac{1}{(k-z+2)i} \right]$$
$$+ \sum_{i=2}^{k+2} \frac{(-1)^i \binom{k+2}{i}\binom{i}{2}}{\lambda} \left[\frac{\exp\left(-\frac{4\lambda c}{k-z}\right)}{i(k-z)+4} - \frac{2\exp\left(-\frac{3\lambda c}{k-z}\right)}{i(k-z)+3} \right].$$

B.2 Proof of Theorem 3.4

For the clarity of presentation, we restate Theorem 3.4.

Theorem (Integral expression leading to $F_{T_{\mathrm{SC}}}(t)$)**.** *The distribution of the master's waiting time T_{SC} of an (n,k,z) system using Staircase codes is given by*

$$F_{T_{SC}}(t) = 1 - n! \int_{(y_k,\dots,y_n)\in\mathcal{A}(t)} \frac{F_{T'}(y_k)^{k-1}}{(k-1)!} dF_{T'}(y_k)\dots dF_{T'}(y_n) \quad \textit{for } t>0. \tag{B.3}$$

We denote the residual service time at each worker $\mathtt{W}_i,\ i=1,\dots,n,$ by the random variable $T'_i = T_i - \frac{c}{k-z}$, and the associated distribution by $F(y_i) \triangleq F_{T'}(y_i) = 1-\exp(-\lambda y_i)$ for $y_i>0$. For $i=k,\dots,n$, we define t_i as $t_i \triangleq \max\left\{\left(\frac{i-z}{k-z}\right)\left(t-\frac{c}{i-z}\right),0\right\}$. We denote by $\mathcal{A}(t)$ the set of ordered variables $(y_k,\dots,y_n)$ such that

$$\mathcal{A}(t) \triangleq \{0 \le y_k \le y_{k+1} \le \cdots \le y_n : t_k < y_k, \dots, t_n < y_n\}.$$

Proof. Let T'_i denote the residual service time of worker i with the offset $\frac{c}{k-z}$. The sequence $(T'_1,\dots,T'_n)$ of residual service times of n workers is assumed to be *iid* and distributed exponentially with rate $\lambda(k-z)$ with the tail-distribution function $\bar{F}_{T'}(t) \triangleq e^{-\lambda(k-z)t}$ for $t>0$.

Since the common distribution of residual service times is absolutely continuous with respect to the Lebesgue measure, the corresponding probability density exists and is denoted by $f_{T'}(t) = dF_{T'}(t)/dt = \lambda(k-z)e^{-\lambda(k-z)t}$ for $t \geqslant 0$. Further, we know that the order statistics $(T'_{(1)},\dots,T'_{(n)})$ of residual times $(T'_1,\dots,T'_n)$ is identical for all their $n!$ permutations. Hence, for any $0 \le y_1 \le \ldots \le y_n$, we can write $f_{T'_{(1)},\dots,T'_{(n)}}(y_1,\dots,y_n) = n! f_{T'_1,\dots,T'_n}(y_1,\dots,y_n) = n!\prod_{i=1}^n f_{T'}(y_i)$. The product form of joint density follows from the independence of the residual service times.

In terms of $\delta_j = \frac{k-z}{j-z}$, the order statistics of residual times $T'_{(j)}$, and the offset $\frac{c}{k-z}$, we can write

$$\{T_{\text{SC}} > t\} = \bigcap_{j=k}^{n} \left\{ T'_{(j)} > \frac{t}{\delta_j} - \frac{c}{j-z} \right\}.$$

For each $k \leq j \leq n$, we define $t_j \triangleq \max\left\{\frac{t}{\delta_j} - \frac{c}{j-z}, 0\right\}, y_{n+1} \triangleq \infty$, and $\hat{A}(t) \triangleq \cap_{j=k}^{n+1}\{t_j < y_j \leq y_{j+1}\} \cap_{j=1}^{k-1} \{0 \leq y_j \leq y_{j+1}\}$. In terms of t_j, y_{n+1} and $\hat{A}(t)$, we can write the tail distribution

$$\Pr\{T_{\text{SC}} > t\} = \int_{y \in \hat{A}(t)} dF_{T'_{(1)},\dots,T'_{(n)}}(y) = n! \int_{t_n}^{\infty} \cdots \int_{t_k}^{y_{k+1}} \prod_{i=k}^{n} dF_{T'}(y_i) \left(\int_0^{y_k} \cdots \int_0^{y_2} \prod_{i=1}^{k-1} dF_{T'}(y_i) \right).$$

First, we compute the integral with respect to ordered non-negative real variables $(y_1, \dots, y_{k-1})$ over the region $B_{k-1} \triangleq \cap_{j=1}^{k-1}\{0 \leq y_j \leq y_{j+1}\}$, a projection of $\hat{A}(t)$ on $(k-1)$ dimensional space spanned by $(y_1, \dots, y_{k-1})$.

Claim B.1. *For each $k > 1$, we have*

$$I_k \triangleq \int_{B_{k-1}} dF_{T'}(y_{k-1}) \dots dF_{T'}(y_1) = \int_0^{y_k} \cdots \int_0^{y_2} \prod_{i=1}^{k-1} dF_{T'}(y_i) = \frac{F_{T'}(y_k)^{k-1}}{(k-1)!}.$$

Proof of Claim B.1. We prove the claim by induction on the number of integration variables k. The base case of $k = 2$ holds trivially true. We assume that the induction hypothesis holds true for some $k \geq 2$, and show that it holds true for $k+1$. This can be shown by writing the integral I_{k+1} in $(k+1)$ integration variables $y_1, \dots, y_{k+1}$ in terms of the integral I_k, and evaluating the integral by substituting the induction hypothesis for I_k as follows

$$I_{k+1} = \int_0^{y_{k+1}} I_k dF_{T'}(y_k) = \int_0^{y_{k+1}} \frac{F_{T'}(y_k)^{k-1}}{(k-1)!} dF_{T'}(y_k).$$

□

Since the projection of $\hat{A}(t)$ on $(n-k+1)$ dimensional space spanned by $(y_k, \dots, y_n)$ is equal to $A(t)$, it follows that the integration of the first part is equal to $n! \int_{(y_k,\dots,y_n) \in A(t)} dF_{T'}(y_n) \dots dF_{T'}(y_k)$, giving us the result. □

Appendix C

Additional Proofs for Chapter 4

C.1 Extension of proof of Theorem 4.1

We prove that the secrecy condition for PRAC given by $H(\mathtt{A}|\mathtt{P}_{\mathcal{Z}}) = H(\mathtt{A})$ is equivalent to proving that $H(\mathtt{R} \mid \mathtt{P}_{\mathcal{Z}}, A) = 0$, where for any subset $\mathcal{Z} \subset \{1, \ldots, n\}, |\mathcal{Z}| = z$, we denote by $\mathtt{P}_{\mathcal{Z}}$ the random variables representing the collection of packets sent to workers indexed by $\mathcal{Z}$. In other words, we need to prove that the random matrices can be decoded given the collection of packets sent to any z workers and the data matrix A. This is the main reason behind encoding the random matrices using an (n, z) MDS code.

We formally prove that $H(\mathtt{R} \mid \mathtt{P}_{\mathcal{Z}}, \mathtt{A}) = 0$ in the proof of Theorem 4.1. Since at each round we generate new random matrices, it is enough to study the privacy condition at one round. Consider a given round t of PRAC. The proof is standard [25, 71, 72] and is given in Lemma 2.4 but we reproduce it here for completeness and to match the notation of PRAC. In what follows, the logarithms in the entropy function are taken base q, where q is a power of prime for which all matrices can be defined in a finite field $\mathbb{F}_q$. We can write,

$$\begin{aligned}
H(\mathtt{A} \mid \mathtt{P}_{\mathcal{Z}}) &= H(\mathtt{A}) - H(\mathtt{P}_{\mathcal{Z}}) + H(\mathtt{P}_{\mathcal{Z}} \mid \mathtt{A}) && \text{(C.1)}\\
&= H(\mathtt{A}) - H(\mathtt{P}_{\mathcal{Z}}) + H(\mathtt{P}_{\mathcal{Z}} \mid \mathtt{A}) - H(\mathtt{P}_{\mathcal{Z}} \mid \mathtt{A}, \mathtt{R}) && \text{(C.2)}\\
&= H(\mathtt{A}) - H(\mathtt{P}_{\mathcal{Z}}) + I(\mathtt{P}_{\mathcal{Z}}; \mathtt{R} \mid \mathtt{A}) && \text{(C.3)}\\
&= H(\mathtt{A}) - H(\mathtt{P}_{\mathcal{Z}}) + H(\mathtt{R} \mid A) - H(\mathtt{R} \mid \mathtt{P}_{\mathcal{Z}}, \mathtt{A}) && \text{(C.4)}\\
&= H(\mathtt{A}) - H(\mathtt{P}_{\mathcal{Z}}) + H(\mathtt{R}) - H(\mathtt{R} \mid \mathtt{P}_{\mathcal{Z}}, \mathtt{A}) && \text{(C.5)}\\
&= H(\mathtt{A}) - z + z - H(\mathtt{R} \mid \mathtt{P}_{\mathcal{Z}}, \mathtt{A}) && \text{(C.6)}\\
&= H(\mathtt{A}) - H(\mathtt{R} \mid \mathtt{P}_{\mathcal{Z}}, \mathtt{A}). && \text{(C.7)}
\end{aligned}$$

Equation (C.2) follows from the fact that given the data A and the keys $R_{t,1}, \dots, R_{t,z}$ all packets generated by the master can be decoded, in particular the packets $P_{\mathcal{Z}}$ received by any z workers can be decoded, i.e., $H(\mathtt{P}_{\mathcal{Z}} \mid \mathtt{A}, \mathtt{R}) = 0$. Equation (C.5) follows because the random matrices are chosen independently from the data matrix A and equation (C.6) follows because PRAC uses z independent random matrices that are chosen uniformly at random from the field $\mathbb{F}_q$. Note from equation (C.5) that for any code to be information theoretically private, $H(\mathtt{R})$ cannot be less then $H(\mathtt{P}_{\mathcal{Z}}) = z$. This means that a secure code must use at least z independent random matrices.

C.2 Proof of Theorem 4.5

For clarity of representation we restate Theorem 4.5.

Theorem. *Let b be the number of row blocks in A, let $\tau_{t,i}$ denote the computation time of the t^{th} packet at worker $\mathtt{W}_i$ and let RTT_i denote the average round-trip time spent to send and receive a packet from worker $\mathtt{W}_i$. The task completion time of PRAC is approximated as*

$$\begin{aligned}
T_{PRAC} &\approx \max_{i \in \{1,\dots,n\}} \{RTT_i\} + \frac{b+\varepsilon}{\sum_{i=z+1}^{n} 1/\mathbb{E}[\tau_{t,i}]}, && \text{(C.8)}\\
&\approx \frac{b+\varepsilon}{\sum_{i=z+1}^{n} 1/\mathbb{E}[\tau_{t,i}]}, && \text{(C.9)}
\end{aligned}$$

where $\mathtt{W}_i$*'s are ordered indices of the workers from fastest to slowest, i.e.,* $\mathtt{W}_1 = \arg\min_i \mathbb{E}[\tau_{t,i}]$.

Proof. The total delay for receiving p_i computed packets from worker $\mathtt{W}_i$ is equal to

$$T_i \approx RTT_i + p_i \mathbb{E}[\tau_{t,i}] \approx p_i \mathbb{E}[\tau_{t,i}]$$

where RTT_i is the average transmission delay for sending one packet to worker $\mathtt{W}_i$ and receiving one computed packet from the worker, $\tau_{t,i}$ is the computation time spent on multiplying packet $P_{t,i}$ by $\mathbf{x}$ at worker $\mathtt{W}_i$, and the average $\mathbb{E}[\tau_{t,i}]$ is taken over all p_i packets. The reason is that PRAC is a dynamic algorithm that sends packets to each worker $\mathtt{W}_i$ with the interval of $\mathbb{E}[\tau_{t,i}]$ between each two consecutive packets and it utilizes the resources of workers fully [178]. The reason behind counting only one round-trip time (RTT) in T_i is that in PRAC, the packets are being transmitted to the workers while the previously transmitted packets are being computed at the worker. Therefore, in the overall delay only one RTT_i is required for sending the first packet $P_{1,i}$ to worker w_i and receiving the last computed packet $P_{p_i,i}\mathbf{x}$ at the master. To approximate the total delay, we assume that the transmission delay of one packet is negligible compared to the computing delay of all p_i packets, which is a valid assumption in practice for IoT-devices at the edge.

On the other hand, in PRAC, the master stops sending packets to workers as soon as it collectively receives $b+\varepsilon$ computed packets from the $n-z$ slowest workers (note that $b+\varepsilon$ is the number of computed packets required for successful decoding, where ε is the overhead due to Fountain Coding), i.e., $\sum_{i=z+1}^{n} p_i = b+\varepsilon$. Note that the z fastest workers are assigned for computing the keys as described in the previous sections. Due to efficiently using the resources of workers by PRAC, all $n-z$ workers will finish computing p_i packets approximately at the same time, i.e., $T_{PRAC} \approx T_i \approx p_i\mathbb{E}[\tau_{t,i}], i = z+1, ..., n$. By replacing p_i with $\frac{T_{PRAC}}{\mathbb{E}[\tau_{t,i}]}$ in $\sum_{i=z+1}^{n} p_i = b+\varepsilon$, we can show that $T_{PRAC} \approx \frac{b+\varepsilon}{\sum_{i=z+1}^{n} 1/\mathbb{E}[\tau_{t,i}]}$. Note that the approximated value approaches the exact value by increasing b. The reason is that the workers' efficiency increases with increasing b. □

C.3 Proof of Theorem 4.6

We restate Theorem 4.6 for the clarity of presentation.

Theorem. *The gap between the completion time of PRAC and coded computation using staircase codes* *is lower bounded by:*

$$\mathbb{E}[T_{SC}] - \mathbb{E}\left[T_{PRAC}\right] \geq \frac{bx - \varepsilon y}{y(x+y)}, \tag{C.10}$$

where $x = \dfrac{n-d^*}{E[\tau_{t,n}]}$, $y = \dfrac{d^*-z}{E[\tau_{t,d^*}]}$ *and* d^* *is the value of* d *that minimizes equation* (4.6)*.*

Proof. We express $\mathbb{E}[T_{\text{SC}}]$ as a function of the computing time $\tau_{t,i}$ of worker $\mathtt{W}_i$, $i = 1, \ldots, n$, as

$$\begin{aligned}\mathbb{E}[T_{\text{SC}}] &= \min_{d \in \{k,\ldots,n\}} \left\{ \frac{k-z}{d-z} \mathbb{E}[T_{(d)}] \right\} &\text{(C.11)}\\ &= \min_{d \in \{k,\ldots,n\}} \left\{ \frac{b}{d-z} \mathbb{E}[\tau_{t,d}] \right\}, &\text{(C.12)}\end{aligned}$$

where $\mathtt{W}_d$ is the d^{th} fastest worker. Next, we find a lower bound on $\mathbb{E}[T_{\text{SC}}] - \mathbb{E}\left[T_{PRAC}\right]$ as follows

$$\begin{aligned}\mathbb{E}[T_{\text{SC}}] - \mathbb{E}\left[T_{PRAC}\right] &= \frac{b}{\frac{d-z}{\mathbb{E}[\tau_{t,d}]}} - \frac{b+\varepsilon}{\sum_{i=z+1}^{n} \frac{1}{\mathbb{E}[\tau_{t,i}]}} &\text{(C.13)}\\ &= \frac{b}{\frac{d-z}{\mathbb{E}[\tau_{t,d}]}} - \frac{b+\varepsilon}{\sum_{i=z+1}^{d} \frac{1}{\mathbb{E}[\tau_{t,i}]} + \sum_{i=d+1}^{n} \frac{1}{\mathbb{E}[\tau_{t,i}]}} &\text{(C.14)}\\ &\geq \frac{b}{\frac{d-z}{\mathbb{E}[\tau_{t,d}]}} - \frac{b+\varepsilon}{(d-z)\frac{1}{\mathbb{E}[\tau_{t,d}]} + (n-d)\frac{1}{\mathbb{E}[\tau_{t,n}]}} &\text{(C.15)}\\ &= \frac{\frac{b(n-d)}{\mathbb{E}[\tau_{t,n}]} - \frac{\varepsilon(d-z)}{\mathbb{E}[\tau_{t,d}]}}{\frac{d-z}{\mathbb{E}[\tau_{t,d}]}\left(\frac{d-z}{\mathbb{E}[\tau_{t,d}]} + \frac{n-d}{\mathbb{E}[\tau_{t,n}]}\right)} &\text{(C.16)}\\ &= \frac{bx - \varepsilon y}{y(x+y)}, &\text{(C.17)}\end{aligned}$$

where $x = \frac{n-d}{\mathbb{E}[\tau_{t,n}]}$ and $y = \frac{d-z}{\mathbb{E}[\tau_{t,d}]}$ and the inequality (C.15) comes from the fact that $z \leq k \leq d \leq n$ and the workers are ordered from the fastest to the slowest. □