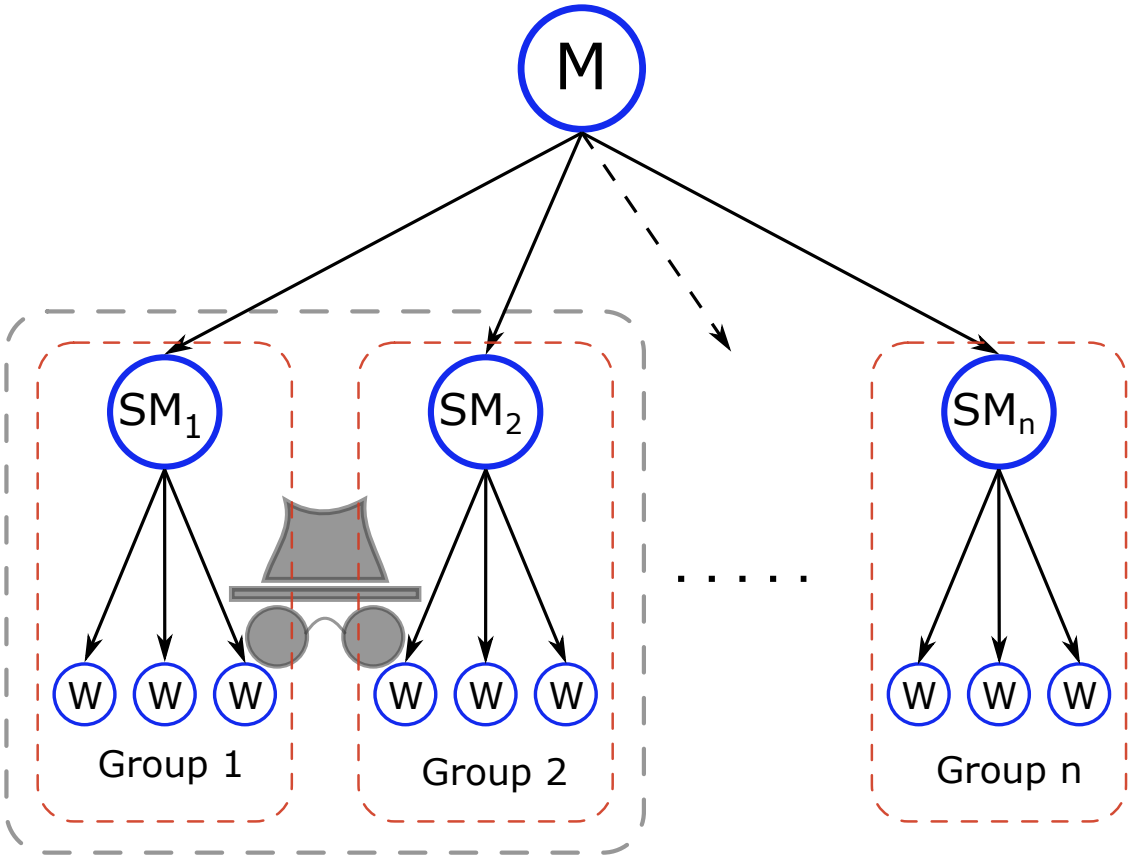
CHALMERS
UNIVERSITY OF TECHNOLOGY



Coding for Distributed Computing

Reed-Solomon Codes for Private, Distributed Computing

Master's Thesis in Communication Engineering

REENT SCHLEGEL

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis EX061/2018

Coding for Distributed Computing

Reed-Solomon Codes for Private, Distributed Computing

REENT SCHLEGEL

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Communication Systems
Chalmers University of Technology
Gothenburg, Sweden 2018

Coding for Distributed Computing
Reed-Solomon Codes for Private, Distributed Computing
REENT SCHLEGEL

Supervisor and examiner:
Alexandre Graell i Amat, Department of Electrical Engineering

Co-supervisor:
Eirik Rosnes, Simula@UiB, Bergen, Norway

Master's Thesis EX061/2018
Department of Electrical Engineering
Division of Communication Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of the considered distributed computing system model.

Typeset in LaTeX
Gothenburg, Sweden 2018

Coding for Distributed Computing
Reed-Solomon Codes for Private, Distributed Computing
REENT SCHLEGEL
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Distributed cluster systems have emerged as a useful way to perform large scale computations. These clusters can be rented at service providers, which imposes constraints regarding the privacy of the data the computations are done on. Furthermore, straggling servers that take an excessive amount of time to finish their computation slow down the overall process gravely. In this thesis, a coding scheme that mitigates the effect of straggling servers and assures privacy is introduced. This is made possible by a concatenation of two Reed-Solomon codes, one for straggler mitigation directly at the service providers and one in combination with random data to assure privacy which is used on the data itself, before it is distributed to multiple service providers. The impact of different network structures on the overall runtime as well as the impact of privacy on the optimal code rate for minimizing the overall runtime is investigated and set in relation with existing theory, such as studies from Lee *et al.* on the optimal code rate of maximum distance separable codes for distributed computing.

Keywords: Coding Theory, Distributed Computing, Information-Theoretic Privacy, Maximum Distance Separable Codes, Private Computing, Reed-Solomon Codes

Acknowledgements

First of all, I would like to thank my supervisors Alexandre Graell i Amat and Eirik Rosnes for the opportunity to work with you. It was an inspiring year and I learned a great deal from both of you. Furthermore, Albin Severinson was always willing to help me with my thesis work. The discussions with you have been very helpful for the deeper understanding of the topic. Thank you!
But I could not have finished this thesis nor my studies in general without the support of my family. Your encouragement to study abroad have definitely influenced my professional education as well as my personal life in a positive way.

Reent Schlegel, Gothenburg, October 2018

Contents

List of Figures **xi**

List of Acronyms **xii**

List of Symbols **xiv**

1 Introduction **1**

2 Preliminaries **3**
2.1 Information Theory 3
2.2 Coding Theory 3
2.2.1 Reed-Solomon Codes 4
2.3 Straggler Problem in Distributed Computing 4

3 System Model **7**
3.1 Computational Task 7
3.2 Architecture 8
3.3 Impact of Adversaries 8

4 Private Scheme **11**
4.1 Block Matrix Form 11
4.2 Encoding 12
4.3 Computational Tasks 12
4.4 Decoding 13

5 Proof of Privacy **15**

6 Code Optimization **17**
6.1 Objective Function 17
6.1.1 Computational Delay 17
6.1.2 Decoding Time 18
6.2 Constraints 19
6.3 Solution 19
6.4 Impact of the Grouping on the Runtime 21

7 Influence of Privacy on the Optimal Code Rate **23**

8 Conclusion and Future Work **27**

Bibliography **29**

List of Figures

2.1 PDF of the computational delay 5

3.1 Architecture of the network of rented servers 8

6.1 Screen shot of the graphical user interface for solving (6.6) 20
6.2 Impact on the runtime of grouping 250 servers 21
6.3 Impact on the runtime of grouping for different numbers of servers . . 22

7.1 Parameters that minimize Lee's objective function for different levels of privacy 24
7.2 Parameters that minimize the overall runtime for different sizes of $\boldsymbol{A}$ 25
7.3 Optimal parameters with adjusted straggling parameters. The tallest matrix acts as reference. 26

List of Acronyms

CDF cumulative distribution function

MDS maximum distance separable

PDF probability density function

SM sub-master

SP service provider

List of Symbols

$\boldsymbol{A}$	matrix with sensitive data
$\tilde{\boldsymbol{A}}$	data matrix
$\hat{\boldsymbol{A}}$	data matrix with randomness
$\boldsymbol{C}$	encoded matrix
$\boldsymbol{C}_i$	code symbol stored at service provider i
$\boldsymbol{C}_{(k)}$	vector of k coded symbols (of fastest k groups)
d	delay of the computational time distribution
d'	scaled delay, $d' = \frac{d}{(k-u)\cdot q}$
$f(t;\mu,d)$	probability density function of computational time
$F(t;\mu,d)$	cumulative distribution function of computational time
$\boldsymbol{E}$	eavesdropped matrix (vector of eavesdropped symbols)
$\mathbb{F}$	finite field the elements of $\boldsymbol{A}$, $\boldsymbol{x}$, $\boldsymbol{R}$, and $\boldsymbol{G}$ are from
$\boldsymbol{G}$	generator matrix of Reed-Solomon code
$\boldsymbol{G}_{(k)}$	$k \times k$ dimensional sub matrix of $\boldsymbol{G}$
k	number of groups to wait for, dimension of inner code
K	number of servers per group, length of outer code
n	number of groups/service providers, length of inner code
N	total number of workers
μ	straggling parameter of the computational time distribution
μ'	scaled straggling parameter, $\mu' = \mu \cdot (k-u) \cdot q$
p	power of the cardinality of $\mathbb{F}$ (2^p is the field size)
q	number of servers to wait for per group, dimension of outer code
r	number of rows in $\boldsymbol{A}$
$\tilde{r}$	number of rows in $\tilde{\boldsymbol{A}}$, divisible by (k-u)
s	number of columns/rows in $\boldsymbol{A}/\boldsymbol{x}$
S_q	qth order statistic of the computational time of the workers
$f_{S_q}(t)$	probability density function of S_q
$F_{S_q}(t)$	cumulative distribution function of S_q
T	computational time
T^{comp}	expected value of computational time
$T^{\text{dec,inner}}$	expected value of decoding time of inner code
$T^{\text{dec,outer}}$	expected value of decoding time of outer code
T_k	kth order statistic of the computational time of the groups
$f_{T_k}(t)$	probability density function of T_k
$F_{T_k}(t)$	cumulative distribution function of T_k
u	number of groups the adversary has access to
$\boldsymbol{x}$	vector for multiplication

1

Introduction

Large scale computations require clusters of servers to run because the burden would be too high for a single machine [1]. There are many service providers (SPs) where such clusters can be rented, such as Google [2], Microsoft [3], and Amazon [4]. This distributed approach entails new challenges. So called stragglers, servers that take an excessive amount of time to finish their computation, delay or even put a stop to the whole computation [5]. One way to avoid stragglers is to invest in expensive hardware which guarantees to fail seldom [1]. Another approach is to design algorithms such that they are resilient against stragglers. Thereby, cheaper hardware, which tends to fail more often, can be used in the computing cluster. Coding theory has been proven to mitigate the effect of straggling servers [6–12]. By adding redundancy to the computations, the system can cope with information losses caused by stragglers. While the opportunities coding theory yields for distributed computing have already been studied for some time and maximum distance separable (MDS) codes have emerged as an efficient way to tackle the straggling problem, another issue remains. When renting servers at a SP, it can not be known if this SP can be trusted.

When performing computations on sensitive data (data that should be kept private) it can happen that the data gets exposed to untrusted parties. As soon as the SP is not honest, all the data is revealed. This leads to the conclusion that private clusters have to be set up when large scale computations on sensitive data are performed. This is very expensive, inconvenient, and often not worth the effort. Instead, it is desired to have a way to perform private computations on rented clusters without the SP gaining any knowledge about the data.

In this thesis, one such scheme is introduced. Coded symbols of a Reed-Solomon code based on data and random source symbols are distributed among multiple SPs. It is shown that the mutual information between the data and the coded symbols stored at a given number of SPs is zero. Thereby, it is guaranteed that a given number of collaborating SPs can not gain any information about the data at all.

The thesis is organized as follows: In Chapter 2, information theory, coding theory, with focus on Reed-Solomon codes and their MDS property, and the straggler problem in distributed computing are introduced. Chapter 3 deals with the system model. Here, the computational task, the architecture of the network of rented servers, and the impact of adversaries are explained. The scheme that assures privacy against the adversaries in the given network is shown in Chapter 4. In Chapter 5, the information-theoretic privacy is proved. In Chapter 6, the various code rates are optimized. The overall expected runtime is minimized for a given computational task, the computational delay distribution of the single servers, and a given number

of collaborating SPs. The impact of privacy and decoding complexity on the optimal code rate is investigated in Chapter 7. Last but not least, the thesis is concluded in Chapter 8.

2
Preliminaries

2.1 Information Theory

Information theory was established by Claude E. Shannon in 1948 in his paper A Mathematical Theory of Communication [13]. Here, an information source is seen as a random variable X. In this work, only discrete sources are treated. A message i from X is a symbol of a given alphabet $\mathcal{I}$ that occurs with a given probability p_i. Then the information in bits of the message is defined as $\log_2(1/p_i)$. Thereby, information can be seen as a measure of uncertainty. The more likely a message is, the less information does it possess and vice versa. The average information of the source is called entropy and is given by

$$H(X) = \sum_{i \in \mathcal{I}} -p_i \log_2(p_i).$$

For a source where every message is equally probable, the entropy corresponds to the logarithm of the number of possible outcomes of this source. Again, the more possible outcomes there are, the higher the uncertainty and with it the entropy.

The conditional entropy $H(X|Y)$ is the average information of a source X, knowing already the message of another source Y. This depends on the statistical dependence of X and Y. For example, if they are independent, $H(X|Y) = H(X)$ holds. The uncertainty about X remains unchanged, because no knowledge about X can be gained by knowledge about Y. On the other hand, when they are fully dependent on each other, for example when they are coupled by a deterministic function, $H(X|Y) = 0$. Knowledge about Y yields total certainty about the outcome of X. Thus, there is no uncertainty about X left.

Last but not least, an important quantity is the mutual information between two random variables, denoted by $I(X;Y)$. It is a measure for the information X and Y yield about each other. It holds that $I(X;Y) = I(Y;X)$ and

$$I(X;Y) = H(X) - H(X|Y).$$

2.2 Coding Theory

Coding theory aims to secure information against errors and erasures; however, in this work only erasures are considered. In a traditional communication system, information shall be transmitted over a noisy or erroneous channel. This is often done in a discrete fashion. The source sequentially produces independent symbols

which form a message. The message is sent over the channel where some of the symbols are altered. In this work, an erasure channel is assumed, where some of the information symbols are lost, but those that are received are not altered in any way.

To avoid information loss, redundancy in the message is introduced. An (n, k) linear code produces n linear combinations of k information symbols. The obtained n symbols are called coded symbols. Naturally, $n > k$ because otherwise no redundancy is introduced. There is a special class of codes, called maximum distance separable (MDS) codes, that have the property that the initial k information symbols can be reconstructed by receiving any k out of the n code symbols. One instance of this class are Reed-Solomon codes.

2.2.1 Reed-Solomon Codes

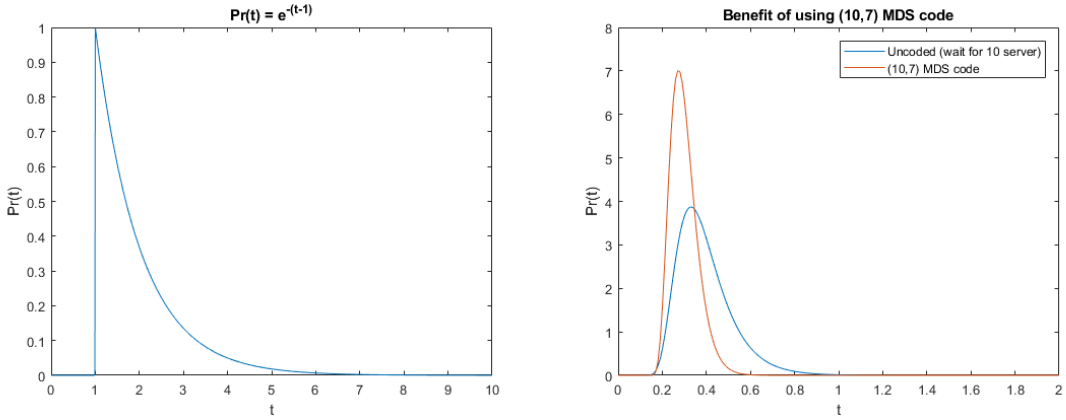
Reed-Solomon codes were introduced in 1960 [14]. They are a well studied example of the MDS codes. They are based on the property of polynomials that every degree $k-1$ polynomial is defined by k evaluation points. A message consisting of k symbols defines coefficients for a degree $k-1$ polynomial. The resulting polynomial is evaluated at n points. By receiving any k out of these n evaluation points, the polynomial, and hence the message can be reconstructed. Let $\boldsymbol{m} = \left[m_0, m_1, ..., m_{k-1}\right]$ be the message and $\boldsymbol{g} = \left[g_0, g_1, ..., g_{n-1}\right]$ the evaluation points. The first coded symbol is obtained as $c_0 = m_0 + m_1 g_0 + ... + m_{k-1} g_0^{k-1}$. As for any linear code, the complete encoding process can be described by a matrix vector multiplication:

$$\boldsymbol{c} = \left[c_0, c_1, ..., c_{n-1}\right] = \left[m_0, m_1, ..., m_{k-1}\right] \cdot \begin{bmatrix} 1 & 1 & \cdots & 1 \\ g_0 & g_1 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_0^{k-1} & g_1^{k-1} & \cdots & g_{n-1}^{k-1} \end{bmatrix} = \boldsymbol{m} \cdot \boldsymbol{G}$$

$\boldsymbol{G}$ is called the generator matrix.
After receiving k coded symbols, the decoding can be done. For this, a system of k linear equations can be obtained, which can be solved by Gaussian elimination with complexity of order $\mathcal{O}(n^3)$. Because of the popularity of Reed-Solomon codes, there has been much research for more efficient decoding algorithms. Berlekamp's algorithm [15] for example and Massey's equivalent algorithm [16] with shift registers allow decoding in $\mathcal{O}(n^2)$ complexity. This algorithm, known as Berlekamp-Massey algorithm, uses the Fourier transform to calculate a syndrome to determine the error and erasure pattern. In [17] a comparative study of the decoding complexity of Gaussian elimination in contrast to Berlekamp-Massey's algorithm is given. In this thesis decoding with Berlekamp-Massey is assumed and the investigation on the decoding complexity is based on [17].

2.3 Straggler Problem in Distributed Computing

Large scale computations that are performed in a distributed manner suffer from the so called straggler effect. When the computational task is split up directly between the workers, the result from all workers is needed to reconstruct the solution to the



(a) One worker **(b)** Distributed over ten workers

Figure 2.1: PDF of the computational delay

computation. For large scale clusters this poses a huge problem. One slow server, a so called straggler, will slow down the whole computation. The computational delay distribution of the single workers usually has a long tail [5]. One such example can be seen in Fig. 2.1a where the probability density function (PDF) of the computational delay of a single server is shown. As the number of servers increases, the likelihood of at least one of them being a straggler rises fast. Without any counter measurements, the overall performance in terms of expected runtime of large scale distributed computations degrades drastically.

Coding theory has been shown to be a useful tool to mitigate the straggler effect [6–12]. By introducing redundancy in the computation, the results of a subset of all servers is sufficient to reconstruct the solution. MDS codes have been shown to be an efficient way of introducing the needed redundancy. By utilizing the k out of n property, up to $n-k$ servers can be stragglers without any performance degradation for the overall system. The workload for every single server is higher compared to uncoded computations (for n servers, the workload is a fraction of $1/k$ instead of $1/n$ of the original computational task) and thereby, the expected computational delay of every server increases as well. But, due to the MDS property, the long tail of the delay distribution can be cut and the overall expected delay is much lower than in the uncoded approach.

This effect is demonstrated in Fig. 2.1b. The expected computational delay of the uncoded approach is 0.393 due to the long tail in the pdf. The coded approach has, regardless of the higher workload per server, an expected delay of 0.299, which is just 76.2% of the uncoded delay. In this case, by utilizing an MDS code, the computation can be speeded up by almost 25%.

3
System Model

In this chapter, the system model is introduced. This includes the computational task that shall be performed, the architecture of the network of rented servers and their computational delay distributions, and the impact of adversaries/non-honest SPs.

3.1 Computational Task

The goal is to perform some linear computation on sensitive data. Linear computations have repeatedly emerged as crucial element in big data analytics. For example, gradient descent as solution algorithm to an optimization problem with quadratic objective function, as often used in machine learning, boils down to a matrix vector multiplication. We will assume a matrix vector multiplication of the form $\boldsymbol{A} \cdot \boldsymbol{x}$, where $\boldsymbol{A} \in \mathbb{F}^{r \times s}$ and $\boldsymbol{x} \in \mathbb{F}^{s}$ have elements of a finite field $\mathbb{F}$, where the cardinality of $\mathbb{F}$ is a power of two, $|\mathbb{F}| = 2^p$. $\boldsymbol{A}$ contains sensible data and is assumed to be huge, such that it is unreasonable to perform this computation locally at the master. Instead, the master will use a number of untrusted SPs to rent servers for the calculation. At each SP a group of workers is rented with an additional sub-master (SM) for this group. Linear combinations of parts of $\boldsymbol{A}$ and random data, which are much smaller than $\boldsymbol{A}$, are distributed to the SMs of each group. The SMs perform calculations on their shares with help from the workers in their group, utilizing an MDS code to minimize the expected computational delay in their group (mitigating the straggler effect), and send back the obtained results to the master, who, in turn, can extract the desired solution from the local results.

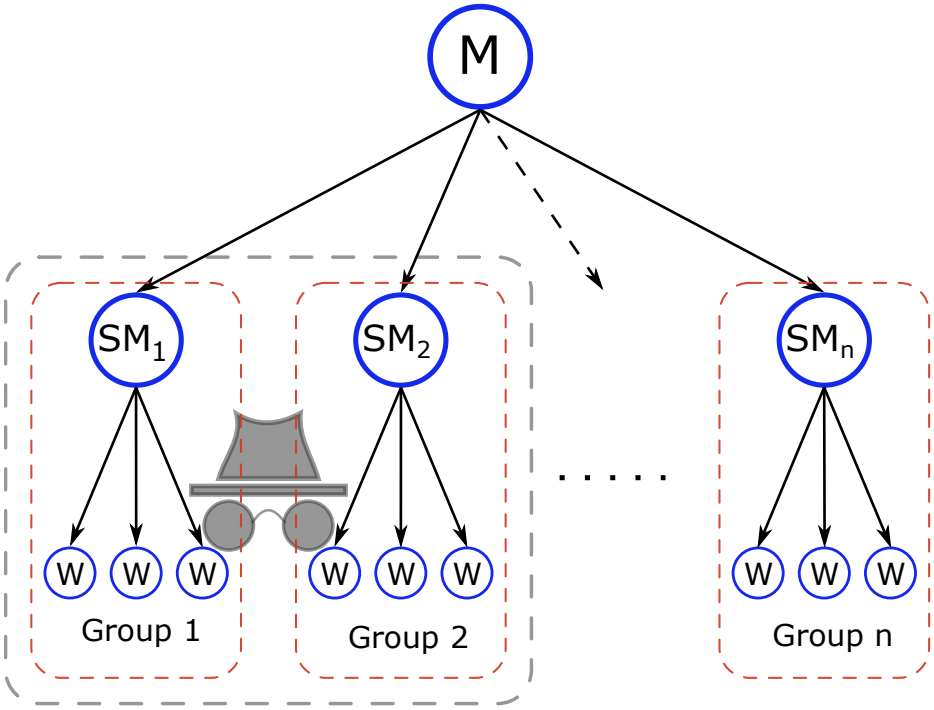


Figure 3.1: Architecture of the network of rented servers

3.2 Architecture

The architecture of the network of rented servers can be seen in Fig. 3.1. The computational task shall be performed with the help of N servers, which are rented at n SPs. This means, there are n groups with $K = N/n$ servers each. It is assumed that all servers have an independent, identically distributed computational delay. The shifted exponential distribution yields a good balance between observations on real server clusters and analytical analyzability [9]. The computational delay T of a single server performing the whole task has a cumulative distribution function (CDF) of the form

$$\Pr(T \leq t) = F(t; \mu, d) = \left(1 - e^{-\mu \cdot (t-d)}\right) \cdot \sigma(t-d), \tag{3.1}$$

with straggling parameter μ, delay d, and heaviside function $\sigma(\tau)$. Hence, the expected computational delay T^{comp} of one server performing the whole task is

$$T^{\text{comp}} = d + \frac{1}{\mu}. \tag{3.2}$$

3.3 Impact of Adversaries

It is assumed that there are SPs that can not be trusted. This means, data that is stored at the groups is exposed to adversaries. Furthermore, it is assumed that multiple SPs work together as one adversary. An adversary has access to the data stored at u SPs. Furthermore, an adversary knows the used scheme. This means, it knows the encoding used for straggler mitigation and privacy assurance.

While the adversary has access to part of the data, it can not influence the computations itself. It is seen as an eavesdropper that just knows what is calculated at the workers, but does not change the outcome of the calculations.

4

Private Scheme

The effect of straggling servers and the presence of adversaries impose challenges for the distributed computing scenario. As coding theory has already shown to provide tools to mitigate the effect of straggling servers, it seems natural to investigate whether the same tools can tackle the second problem: providing privacy against adversaries.

In this chapter, a privacy assuring coding scheme is introduced. A concatenation of two Reed-Solomon codes provides resilience against straggling servers as well as straggling groups of servers and assure privacy against a number of collaborating adversaries. An outer (n, k) Reed-Solomon code is used over the groups to mitigate the effect of straggling groups and provide privacy. An inner (K, q) code mitigates the effect of straggling servers. How to choose the code parameters n, k, K, and q will be explained in Chapter 6.

First, $\boldsymbol{A}$ has to be transformed into a block-matrix. These blocks are taken together with random matrices as source symbols for the outer code. The obtained code symbols are sent to the SPs, which, in turn, perform calculations on the coded symbols utilizing the K servers in their group. In the groups, the inner (K, q) code is used to mitigate the effect of straggling servers by waiting for just the fastest q servers in the group to obtain the local result. After obtaining the k fastest results from the groups, the master can extract the desired solution $\boldsymbol{Ax}$.

4.1 Block Matrix Form

In order for all block matrices to have the same size, matrix $\boldsymbol{A}$ has to be zero padded. This is done by appending r mod $(k-u)$ zero rows to $\boldsymbol{A}$. The newly obtained structure has the form

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{0} \end{bmatrix} \in \mathbb{F}^{\tilde{r} \times s},$$

where $k-u$ divides $\tilde{r}$.
To conceal any information about $\boldsymbol{A}$, randomness in the data sent to the groups is needed. This is done by generating a random matrix $\boldsymbol{R} \in \mathbb{F}^{\frac{u \cdot \tilde{r}}{k-u} \times s}$ and forming

$$\hat{\boldsymbol{A}} = \begin{bmatrix} \tilde{\boldsymbol{A}} \\ \boldsymbol{R} \end{bmatrix}.$$

$\hat{\boldsymbol{A}}$ can be seen as a $k \times 1$ block matrix, where each block is in $\mathbb{F}^{\frac{\tilde{r}}{k-u} \times s}$:

$$\hat{\boldsymbol{A}} = \begin{bmatrix} \tilde{\boldsymbol{A}}_1 \\ \tilde{\boldsymbol{A}}_2 \\ \vdots \\ \tilde{\boldsymbol{A}}_{k-u} \\ \boldsymbol{R}_1 \\ \vdots \\ \boldsymbol{R}_u \end{bmatrix}$$

These k blocks can be taken as source symbols of an (n, k) Reed-Solomon code.

4.2 Encoding

The encoding is done with help of a generator matrix $\boldsymbol{G}$ of a Reed-Solomon code as introduced in Section 2.2.1. $\boldsymbol{G}$ is a $k \times n$ scalar matrix, that defines coefficients for the linear combination of the message symbols to obtain the coded symbols. In our case, the symbols are the blocks of $\hat{\boldsymbol{A}}$. The (block-)column of $\hat{\boldsymbol{A}}$ can be seen as a message, that shall be encoded with $\boldsymbol{G}$. Hence, the encoding can be written as $\hat{\boldsymbol{A}}^T \cdot (\boldsymbol{G} \otimes \boldsymbol{I}_{\frac{\tilde{r}}{k-u}})$, where $\otimes$ represents the Kronecker product and $\boldsymbol{I}_{\frac{\tilde{r}}{k-u}}$ is the identity matrix of same dimension as the number of rows in the blocks of $\hat{\boldsymbol{A}}$. By transposing this equation, the encoded matrix $\boldsymbol{C}$ is obtained as:

$$\boldsymbol{C} = (\boldsymbol{G}^T \otimes \boldsymbol{I}_{\frac{\tilde{r}}{k-u}}) \cdot \hat{\boldsymbol{A}},$$

where the columns of $\boldsymbol{C}$ are a codeword of the Reed-Solomon code.

4.3 Computational Tasks

The encoded matrix $\boldsymbol{C}$ can be written in block form as well:

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_1 \\ \vdots \\ \boldsymbol{C}_n \end{bmatrix}$$

Now every SP i gets $\boldsymbol{C}_i$ to calculate $\boldsymbol{C}_i \cdot \boldsymbol{x}$ in its group. First, to mitigate the effect of straggling servers, the SM applies a (K, q) Reed-Solomon code on C_i and sends the coded symbols to the K workers. Then, the servers in the group perform the calculation on their share and send back the result as soon as they competed their task. After receiving the result of the q fastest servers, the SM can extract the solution to $C_i \cdot x$ from the q intermediate results and send this back to the master.

4.4 Decoding

The master waits for the fastest k groups to finish their computation and stores the results in a vector of the form

$$\begin{bmatrix} \boldsymbol{C}_{i_1} \cdot \boldsymbol{x} \\ \boldsymbol{C}_{i_2} \cdot \boldsymbol{x} \\ \vdots \\ \boldsymbol{C}_{i_k} \cdot \boldsymbol{x} \end{bmatrix} = \boldsymbol{C}_{(k)} \cdot \boldsymbol{x}, \text{ with } i_1 < i_2 < ... < i_k.$$

The encoding matrix of the Reed-Solomon code is a Vandermonde matrix and every quadratic sub-matrix of dimension k, $\boldsymbol{G}_{(k)}$, has full rank. Because of this, it is invertible and decoding can be done with the inverse of $\boldsymbol{G}_{(k)}$. Assume $\boldsymbol{G}_{(k)}$ is the sub-matrix of $\boldsymbol{G}$ with columns corresponding to $i_1, i_2, ..., i_k$. Then it holds that $\boldsymbol{C}_{(k)} = (\boldsymbol{G}_{(k)}^T \otimes \boldsymbol{I}_{\frac{\tilde{r}}{k-u}}) \cdot \hat{\boldsymbol{A}}$ and the decoding can be done as follows:

$$((\boldsymbol{G}_{(k)}^T)^{-1} \otimes \boldsymbol{I}_{\frac{\tilde{r}}{k-u}}) \cdot \boldsymbol{C}_{(k)} \cdot \boldsymbol{x} = ((\boldsymbol{G}_{(k)}^T)^{-1} \otimes \boldsymbol{I}_{\frac{\tilde{r}}{k-u}}) \cdot (\boldsymbol{G}_{(k)}^T \otimes \boldsymbol{I}_{\frac{\tilde{r}}{k-u}}) \cdot \hat{\boldsymbol{A}} \cdot \boldsymbol{x} = \hat{\boldsymbol{A}} \cdot \boldsymbol{x} = \begin{bmatrix} \boldsymbol{A} \cdot \boldsymbol{x} \\ \boldsymbol{0} \\ \boldsymbol{R} \cdot \boldsymbol{x} \end{bmatrix}$$

The first r rows are the desired result.

5
Proof of Privacy

In this chapter, the information theoretic privacy of the proposed scheme is proved. Thereby, it is assured that up to u coded symbols give no information about the sensitive data. In other words, it is shown that up to u collaborating SPs gain no knowledge about $\boldsymbol{A}$.

Due to the way encoding is done, the (block-)column of $\boldsymbol{C}$ is a codeword of length n of the used Reed-Solomon code. Since $\boldsymbol{C}$ is row partitioned before it is sent to the groups, every group receives exactly one out of the n symbols of codeword $\hat{\boldsymbol{A}}$ was encoded in.

Assume an adversary has access to v groups, denoted by $i_1, \ldots, i_v$. It gains knowledge about

$$\left[\boldsymbol{C}_{i_1}, \quad \cdots, \quad \boldsymbol{C}_{i_v}\right] = \boldsymbol{E},$$

where $\boldsymbol{E}$ is the vector of eavesdropped symbols of $\boldsymbol{C}$. For the scheme to be private, the mutual information between $\boldsymbol{A}$ and $\boldsymbol{E}$ must be zero:

$$I(\boldsymbol{A}; \boldsymbol{E}) \stackrel{!}{=} 0.$$

As in [18] it holds:

$$\begin{aligned}
I(\boldsymbol{A}; \boldsymbol{E}) &= H(\boldsymbol{A}) - H(\boldsymbol{A}|\boldsymbol{E}) \\
&\stackrel{(a)}{=} H(\boldsymbol{A}) - H(\boldsymbol{A}|\boldsymbol{E}) + H(\boldsymbol{E}|\boldsymbol{A}, \boldsymbol{R}) \\
&= H(\boldsymbol{A}) - H(\boldsymbol{A}|\boldsymbol{E}) + H(\boldsymbol{E}|\boldsymbol{A}) - I(\boldsymbol{R}, \boldsymbol{E}|\boldsymbol{A}) \\
&\stackrel{(b)}{=} H(\boldsymbol{E}) - I(\boldsymbol{R}, \boldsymbol{E}|\boldsymbol{A}) \\
&= H(\boldsymbol{E}) - H(\boldsymbol{R}|\boldsymbol{A}) + H(\boldsymbol{R}|\boldsymbol{E}, \boldsymbol{A}) \\
&\stackrel{(c)}{=} H(\boldsymbol{E}) - H(\boldsymbol{R}) + H(\boldsymbol{R}|\boldsymbol{E}, \boldsymbol{A}),
\end{aligned}$$

where (a) follows from $H(\boldsymbol{E}|\boldsymbol{A}, \boldsymbol{R}) = 0$, because $\boldsymbol{E}$ is a function of $\boldsymbol{A}$ and $\boldsymbol{R}$, (b) from $H(\boldsymbol{E}) - H(\boldsymbol{E}|\boldsymbol{A}) = H(\boldsymbol{A}) - H(\boldsymbol{A}|\boldsymbol{E})$, and (c) from the fact, that $\boldsymbol{R}$ and $\boldsymbol{A}$ are stochastically independent ($H(\boldsymbol{R}|\boldsymbol{A}) = H(\boldsymbol{R})$). This means,

$$I(\boldsymbol{A}; \boldsymbol{E}) = 0 \Leftrightarrow H(\boldsymbol{R}|\boldsymbol{E}, \boldsymbol{A}) = H(\boldsymbol{R}) - H(\boldsymbol{E})$$

$\boldsymbol{R}$ consist of u blocks with $\frac{\tilde{r} \cdot s}{k-u}$ elements from $\mathbb{F}$ which take one of 2^p different, uniformly distributed values each. This yields

$$H(\boldsymbol{R}) = \log_2\left(\left(2^p\right)^{\frac{u \cdot \tilde{r} \cdot s}{k-u}}\right) = \frac{p \cdot u \cdot \tilde{r} \cdot s}{k-u}. \tag{5.1}$$

Similarly it holds for $\boldsymbol{E}$

$$H(\boldsymbol{E}) = \log_2\left((2^p)^{\frac{v\cdot\tilde{r}\cdot s}{k-u}}\right) = \frac{p\cdot v\cdot\tilde{r}\cdot s}{k-u}. \tag{5.2}$$

Now, $H(\boldsymbol{R}|\boldsymbol{E},\boldsymbol{A})$ has to be determined. For this, $\boldsymbol{E}$ can be written as

$$\boldsymbol{E} = \hat{\boldsymbol{A}}^T\cdot\left(\boldsymbol{G}_{(v)}\otimes\boldsymbol{I}_{\frac{\tilde{r}}{k-u}}\right), \tag{5.3}$$

where $\boldsymbol{G}_{(v)}$ is the sub matrix of $\boldsymbol{G}$ with columns corresponding to $i_1,\ldots,i_v$. For a single (block-)element in $\boldsymbol{E}$ holds with (5.3):

$$\boldsymbol{E}_j = \hat{\boldsymbol{A}}^T\cdot\left(\boldsymbol{G}_{(i_j)}\otimes\boldsymbol{I}_{\frac{\tilde{r}}{k-u}}\right) = \sum_{l=1}^{k-u}\boldsymbol{A}_l^T\cdot G_{l,i_j} + \sum_{l=1}^{u}\boldsymbol{R}_l^T\cdot G_{l+k-u,i_j}, \tag{5.4}$$

From now on, it is assumed that $\boldsymbol{E}$ and $\boldsymbol{A}$ are known. Every element in $\boldsymbol{E}_j$ can be seen as the weighted summation of u unknown elements (elements of $\boldsymbol{R}$) of the field $\mathbb{F}$ and there are exactly v of these sums. This means, if $v\leq k$, for every element in $\boldsymbol{E}_j$ there is a system of v linearly independent equations and u unknowns. Note, that due to the MDS property of the Reed-Solomon code, all k distinct code symbols are linearly independent. Hence, all $v\leq k$ distinct equations of the form of (5.4) are linearly independent. Because of this, as long as $v\leq u$, there remain $(2^p)^{(u-v)}$ possible solutions for every position in the blocks of $\boldsymbol{R}$. There are $\frac{\tilde{r}\cdot s}{k-u}$ such positions. This leads to:

$$H(\boldsymbol{R}|\boldsymbol{E},\boldsymbol{A}) = \log_2\left((2^p)^{\frac{(u-v)\cdot\tilde{r}\cdot s}{k-u}}\right) = \frac{p\cdot(u-v)\cdot\tilde{r}\cdot s}{k-u} \tag{5.5}$$

With (5.1), (5.2), and (5.5) it can be seen that $H(\boldsymbol{R}|\boldsymbol{E},\boldsymbol{A}) = H(\boldsymbol{R}) - H(\boldsymbol{E})$ and hence, $I(\boldsymbol{A};\boldsymbol{E}) = 0$. This holds true as long as $v\leq u<k$. Only then there are $(2^p)^{(u-v)}$ possible solutions per entry to (5.4).

6
Code Optimization

In this chapter, the proposed scheme (i.e. the code rates) is optimized with respect to a minimal expected runtime. Furthermore, a sensitivity analysis of the solution is made. First, the optimization problem is formulated. The objective function and the feasible set are introduced. Then, a solution algorithm based on an exhaustive search is explained. The fact that the feasible set is discrete and has a small cardinality makes it possible to try all combinations of code parameters and search for the one yielding the smallest expected runtime. In the end, the impact on the runtime of different groupings of the servers to the SPs is investigated.

6.1 Objective Function

The objective is to minimize the overall expected waiting time. This consists of the expected computational delay at the groups and the decoding time at the SM and at the master (decoding of inner and outer code). This means, the objective is:

$$\text{minimize} \quad T = T^{\text{comp}} + T^{\text{dec,inner}} + T^{\text{dec,outer}} \tag{6.1}$$

In the following subsections, the single composites of T are explained in detail.

6.1.1 Computational Delay

Every SP receives one code symbol of the outer (n, k) Reed-Solomon code. By utilizing an inner (K, q) Reed-Solomon code for straggler mitigation, the matrix vector multiplication of the coded symbol and $\boldsymbol{x}$ shall be performed. Thus, the computational load for every worker is $\frac{1}{k \cdot q}$ of the computational load of performing $\hat{\boldsymbol{A}} \cdot \boldsymbol{x}$. Furthermore, the computational load to compute $\hat{\boldsymbol{A}} \cdot \boldsymbol{x}$ compared to compute $\boldsymbol{A} \cdot \boldsymbol{x}$ scales as $\frac{k}{k-u}$. Combining these factors yields that the computational load for every worker is a fraction of $\frac{1}{(k-u) \cdot q}$ compared to performing the initial task $\boldsymbol{A} \cdot \boldsymbol{x}$. In (3.1) the CDF of the computational delay $F(t; mu, d)$ is given for one worker performing the whole task. By applying the introduced scheme, a new, scaled version is obtained with scaled straggling parameter $\mu' = \mu \cdot (k-u) \cdot q$ and delay $d' = \frac{d}{(k-u) \cdot q}$:

$$\begin{aligned} F((k-u) \cdot q \cdot t; \mu, d) &= \left(1 - e^{-\mu \cdot ((k-u) \cdot q \cdot t - d)}\right) \cdot \sigma((k-u) \cdot q \cdot t - d) \\ &= \left(1 - e^{-\mu \cdot (k-u) \cdot q \cdot (t - \frac{d}{(k-u) \cdot q})}\right) \cdot \sigma(t - \frac{d}{(k-u) \cdot q}) \\ &= \left(1 - e^{-\mu' \cdot (t - d')}\right) \cdot \sigma(t - d') = F(t; \mu', d') \end{aligned}$$

Due to the MDS property of the used inner code, the computation in the group is finished after the first q workers are done with their calculation. The qth smallest out of K realizations of a random variable is called the qth order statistic. The PDF $f_{S_q}(t)$ of the qth order statistic of a random variable with CDF $F(t;\mu,d)$ and PDF $f(t;\mu,d)$ is given by [19]:

$$f_{S_q}(t) = \frac{K!}{(q-1)!(K-q)!} \cdot F(t;\mu,d)^{q-1} \cdot \big(1 - F(t;\mu,d)\big)^{K-q} \cdot f(t;\mu,d) \tag{6.2}$$

This means, the time one group needs to finish their task has a PDF $f_{S_q}(t)$ according to (6.2) and a CDF

$$F_{S_q}(t) = \int_{-\infty}^{t} f_{S_q}(\tau) \ \ \mathrm{d}\tau.$$

The overall computation is done after the first k groups are done with their computations (and decoding, more in Sections 6.1.2 and 6.3). This means, the computational delay is the kth order statistic of S_q and has the PDF

$$f_{T_k}(t) = \frac{n!}{(k-1)!(n-k)!} \cdot F_{S_q}(t)^{k-1} \cdot \big(1 - F_{S_q}(t)\big)^{n-k} \cdot f_{S_q}(t).$$

Finally, the overall expected computational delay is obtained as

$$T^{\text{comp}} = \int_{-\infty}^{\infty} t \cdot f_{T_k}(t) \ \ \mathrm{d}t \tag{6.3}$$

with $F(t;\mu',d')$ as CDF of the computational delay of the single workers.

6.1.2 Decoding Time

There are two codes to decode. First, the SM in each group has to decode the inner code after the fastest q workers have finished their calculations. Secondly, the master has to decode the outer code after receiving the fastest k results from the groups. In both cases a Reed-Solomon code is used. Hence, they have the same structure and decoding procedure but differ in their dimensions. Therefore, the derivation of the decoding time will be done once for the outer (n,k) code and then the decoding time of the inner code is obtained by substituting the dimensions to (K,q).

The initial matrix vector multiplication takes $\frac{rsk}{k-u}$ multiplications and $\frac{r(s-1)k}{k-u}$ additions and the expected computational time is according to (3.2) $T^{\text{comp}} = d + 1/\mu$. Assume one addition takes t_{a} time and one multiplication takes $\log(p)t_{\mathsf{a}}$ time. Than one obtains on average for t_{a}:

$$\frac{rsk}{k-u} \cdot \log(p)t_{\mathsf{a}} + \frac{r(s-1)k}{k-u} \cdot t_{\mathsf{a}} = d + \frac{1}{\mu} \quad \Leftrightarrow \quad t_{\mathsf{a}} = \frac{(d+1/\mu)\cdot(k-u)}{rk(s\log(p)+s-1)}$$

Decoding a (n,k) Reed-Solomon code with the Berlekamp-Massey algorithm takes $n(n-k-1)$ multiplications and $n(n-k)$ additions [17]. Hence,

$$\begin{aligned} T^{\text{dec,outer}} &= n(n-k-1)\cdot\log(p)t_{\mathsf{a}} + n(n-k)\cdot t_{\mathsf{a}} \\ &= n\big((n-k-1)\cdot\log(p) + n - k\big) \cdot \frac{(d+1/\mu)\cdot(k-u)}{rk(s\log(p)+s-1)}, \end{aligned} \tag{6.4}$$

and

$$T^{\text{dec,inner}} = K\big((K-q-1)\cdot\log(p)+K-q\big)\cdot\frac{(d+1/\mu)\cdot(k-u)}{rk(s\log(p)+s-1)} \tag{6.5}$$

respectively. With (6.1), (6.3), (6.4), and (6.5) the objective function is well defined.

6.2 Constraints

There are a couple of fixed parameters. Some are given by the architecture, such as the computational delay distribution, or rather the straggling parameter μ and delay d, some by the computational task, such as the matrix dimension r and s and the field size through p, and finally, some are given by the network design, such as the number of rented servers N and the security level u.

Moreover, there are the variables that can be tweaked to minimize the overall runtime. These are the code dimensions (n,k) and (K,q) respectively. These variables are subject to specific constraints for the scheme to remain feasible. For example, they all have to be natural numbers larger than zero. Then, it is assumed that all groups have the same size K and that all N servers are assigned to a group. Hence, the number of groups n must be a divisor of the number of servers N and every group gets $K = N/n$ servers assigned. Furthermore, the code rates n/k and K/q are upper bounded by 1. Lastly, since the desired result can be reconstructed with the responses of any k groups, the number of SPs the adversary has access to must be strictly less than k. Otherwise the adversary could reconstruct the result as well. All these constraints can be expressed as a mathematical model. Variables that fulfill the constraints form the so called feasible set. The feasible set together with the objective function compose an optimization problem given in (6.6).

$$\begin{aligned}
\underset{n,k,K,q}{\text{minimize}} \quad & T = T^{\text{comp}} + T^{\text{dec,inner}} + T^{\text{dec,outer}} \\
\text{subject to} \quad & n \in \{n : N \bmod n = 0\}, \\
& u < k \leq n, \\
& K = N/n, \\
& 0 < q \leq K, \\
& n,k,K,q \in \mathbb{N}
\end{aligned} \tag{6.6}$$

6.3 Solution

There is a straightforward numerical solution to (6.6). Since the feasible set is discrete and has a cardinality of just $\sum_{n\in\{n:N \bmod n=0 \,\wedge\, n>u\}}(n-u)\cdot K$, an exhaustive search for the optimal code dimensions is possible. For all feasible groupings, n and K, it is possible to go through all dimensions of the outer code k; those are in the range of $u+1$ till n. With the given K, μ', and d' the inner code can be optimized. For every q in the range from 1 to K, T^{comp} can be computed according to (6.3). Then $T^{\text{dec,inner}}$ is added to T^{comp}. The q, for which this sum is the smallest, is the

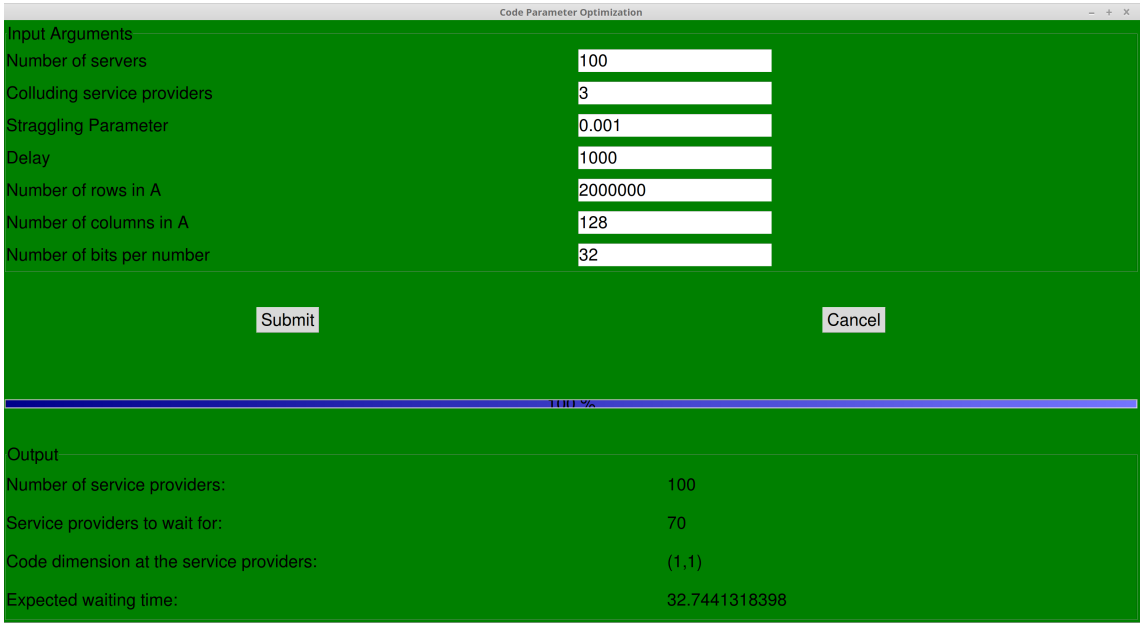


Figure 6.1: Screen shot of the graphical user interface for solving (6.6)

optimal inner code dimension for the given K, k, and u. For every k, the smallest sum is stored. To this, the decoding time of the outer code is added. The smallest of these sums is the optimal expected runtime for the given grouping n. After this is repeated for every feasible grouping, the smallest expected runtime can be obtained and hence, the solution to (6.6).

Within the framework of this thesis, a graphical user interface that solves (6.6) for given μ, d, r, s, p, N, and u was developed. It is available in [20]. A screen shot of the interface can be seen in Fig. 6.1. With help of this program, it can be seen that it is preferable to spread the servers over as many SPs as possible. The optimal runtime is achieved if one server is rented per SP. This is somewhat intuitive because the number of collaborating SPs u is kept constant. The more the information is spread, the harder it is for a fixed number of groups to gain knowledge about the information. Thereby, less random data to obscure information is needed and, in turn, the computational load is lower. This positively affects the runtime and is only opposed by the higher decoding effort that is needed for longer codes. While the inner code is very short, or rather non existent, the outer code is extremely long. Due to the quadratic nature of the decoding complexity two medium long codes would be preferable. But apparently the benefit of less computational load outweighs the increased decoding complexity.

There remains the problem that renting a small amount of servers at many SPs may be unreasonable to do. For example, it will be cumbersome, hence expensive because it takes more organizing and time, to rent one server at 100 SPs. Rather, it is preferred to rent ten servers at ten SPs. This raises the question how the overall runtime is affected by different groupings, which is addressed in the next section.

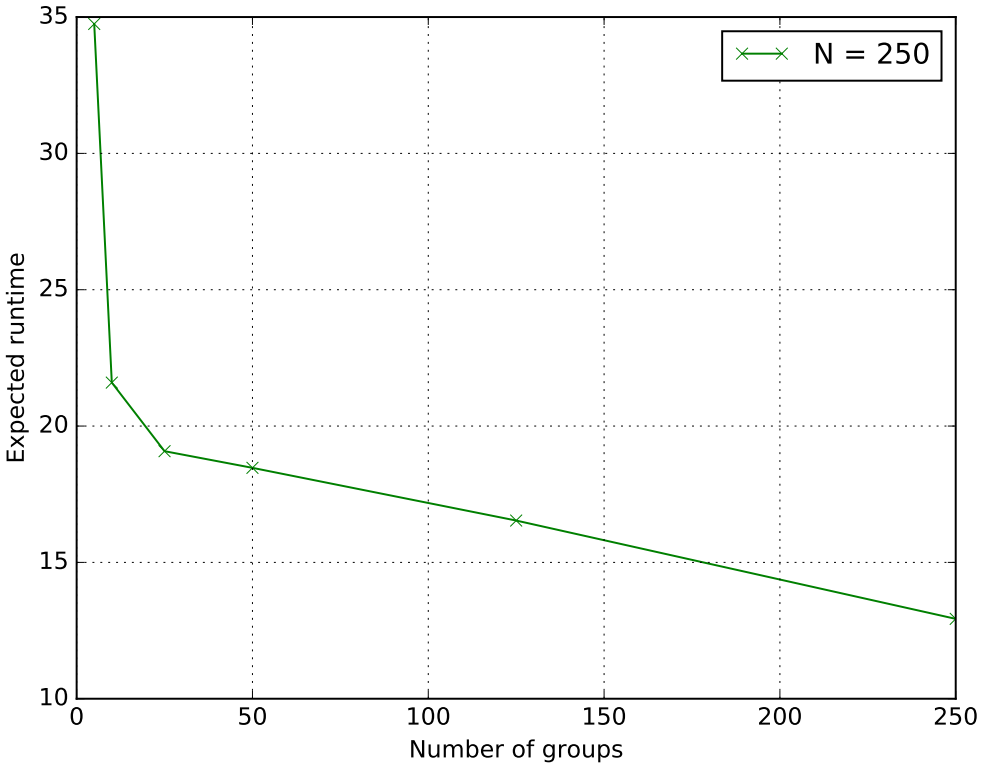


Figure 6.2: Impact on the runtime of grouping 250 servers

6.4 Impact of the Grouping on the Runtime

As described in Section 6.3, there are multiple ways to group the N rented servers; in other words: There are a lot of possible realizations of n, the number of used SPs. While the solution which utilizes the most SPs is optimal in meanings of the overall runtime, it will seldom be the cheapest or easiest to implement. That is why it can be interesting to see which impact the grouping has on the overall runtime. Thereby, the performance degradation can be estimated to trade off the additional cost by using more SPs.

In Fig. 6.2 the overall runtime for $N = 250$, $u = 3$, $\mu = 0.001$, $d = 1000$, $r = 2000000$, $s = 128$, and $p = 32$ is shown. Here, a clear benefit results from using as much SPs as possible. There is a reduction in the runtime of 22% by using 250 instead of 125 SPs. But when more servers are used, this holds not true anymore. Fig. 6.3 shows the impact of grouping on the runtime for multiple N. With $N = 1000$ servers, there is no notable benefit of using 1000 SPs over 20 SPs.

This leads to the conclusion that it is definitely worth it to check the behavior of the runtime in dependence of the grouping because the trade off against the additional efforts and costs of using more SPs may turn out in favor for much fewer groups. The code to generate the plots is based on the same solution algorithm as in the graphical user interface introduced in Section 6.3 and can be found in [20] as well.

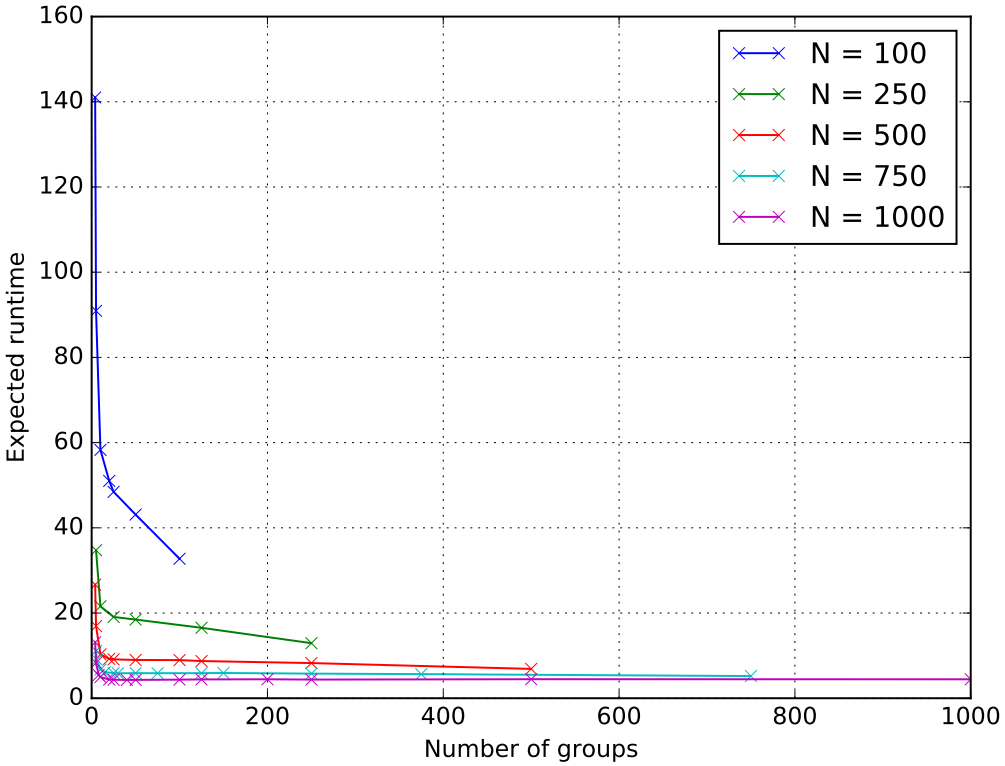


Figure 6.3: Impact on the runtime of grouping for different numbers of servers

7

Influence of Privacy on the Optimal Code Rate

MDS codes have been shown to mitigate the straggler effect. While increasing the computational load for every worker compared to the uncoded scheme, they cut off the tail of the latency distribution and thereby the overall runtime is decreased. The smaller the dimension of the code is, the more each worker has to compute which increases the single computational delays. Simultaneously, the results of fewer workers are sufficient to reconstruct the solution to the initial problem. This raises the question of which code dimension yields the minimal overall expected runtime.

Lee *et al.* minimized in [9] the computational delay for various coding schemes, including MDS codes and obtained an expression for the optimal code rate for MDS codes depending on the straggling parameter μ. It is interesting to see how privacy constraints influence these rates. Lee *et al.* used only one code instead of two as in the scheme in this thesis. But since the inner code is non existent in the optimal case ($K = q = 1$), the two schemes are identical in this regard. The objective to minimize in [9] is the computational delay of the fastest k out of n servers. In this thesis, the decoding time was taken into account as well. The generality of Lee's study did not allow for the inclusion of the decoding complexity of a specific code. That is why the investigation on the optimal code rate will be done in two steps: First, the influence of adding privacy constraints to the computation problem is analyzed for Lee's objective function. Then, in the second step, the objective is changed to the one used in this thesis, which additionally takes the decoding time into account as introduced in Section 6.1.

In Fig. 7.1 the code rate that minimizes the computational delay for a specific straggling parameter and the resulting time can be seen for different levels of privacy. In this case, normalized privacy means the number of collaborating SPs u divided by the total number of used SPs n. Lee's original values form the blue curves corresponding to a normalized privacy of zero. As can be seen, privacy constraints favor a higher code rate and slow down the overall computation. Even for very low straggling parameters, which correspond to a long tail in the latency distribution, the code rate does not fall below 0.4 for a normalized privacy of 10%. Interestingly, the constraints from Section 6.2 demand only $k > u$ which suggests that the optimal code rate for low μ settles at 0.1 rather than 0.4. Hence, there must be another explanation for the observed increase in the code rate.

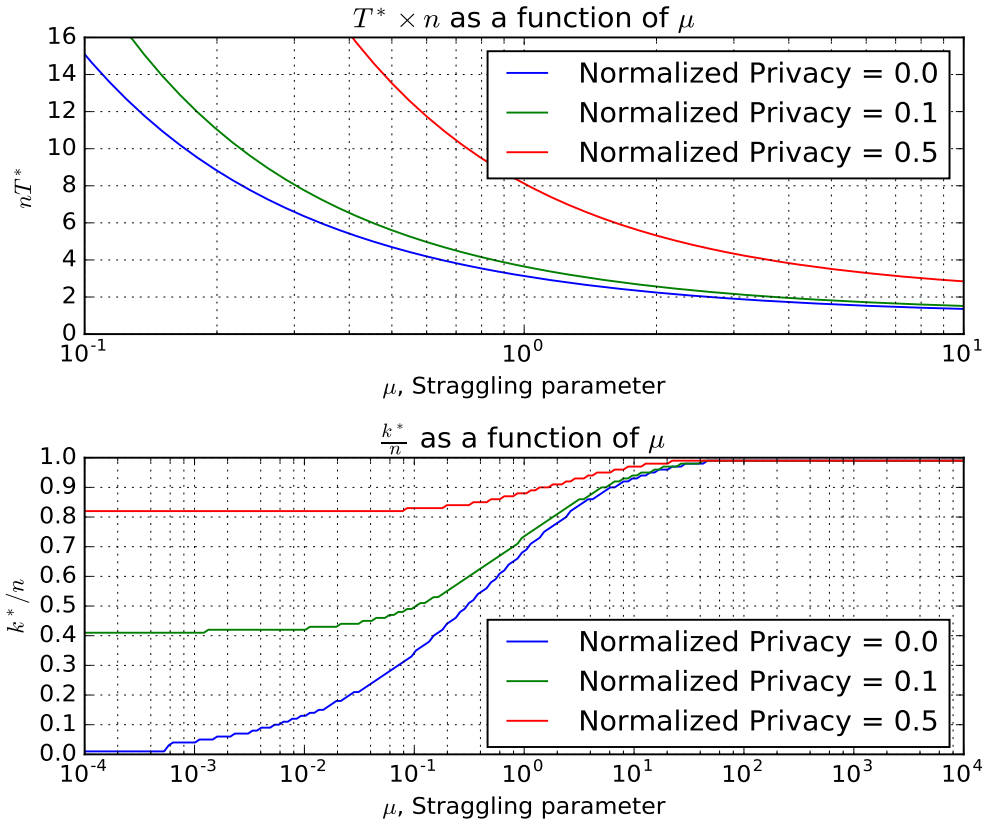


Figure 7.1: Parameters that minimize Lee's objective function for different levels of privacy

The additional computational load imposed by the privacy constraints scales relatively to the initial computational load as $\frac{k}{k-u} - 1$. For fixed u, $\lim_{k\to\infty} \frac{k}{k-u} - 1 = 0$. Hence, a high code rate reduces the additional computational load and thereby the expected computational delay is reduced as well. That is why an increase in the code rate beyond the constraint $k > u$ is observed.

A further increase in the code rate can be observed when the decoding time is taken into account as well. As shown in Section 6.1.2, a higher code rate reduces the complexity of decoding. Fig. 7.2 shows the optimal code rates depending on μ for a normalized privacy of 0.1 and different sizes of the matrix $\boldsymbol{A}$. To quantify the decoding complexity, the code length n has to be set. Here, $n = 100$ which yields $u = 10$. As can be seen, for a large matrix the impact of decoding is marginal. Compared to the initial computation, the decoding process is negligible. For a comparatively small matrix, the decoding complexity has a noticeable impact. Hence, a higher code rate is favored, as expected. Strangely, the computations on the small matrix take more time than on the big matrix. This is due to the effect that direct comparison between the curves in Fig. 7.2 is not fair. The straggling parameter is kept constant while the matrix sizes change. Hence, for the smaller matrices slower servers are used for the calculation. Because of this, the runtime is higher. To make a fair comparison, the straggling parameter has to be adapted.

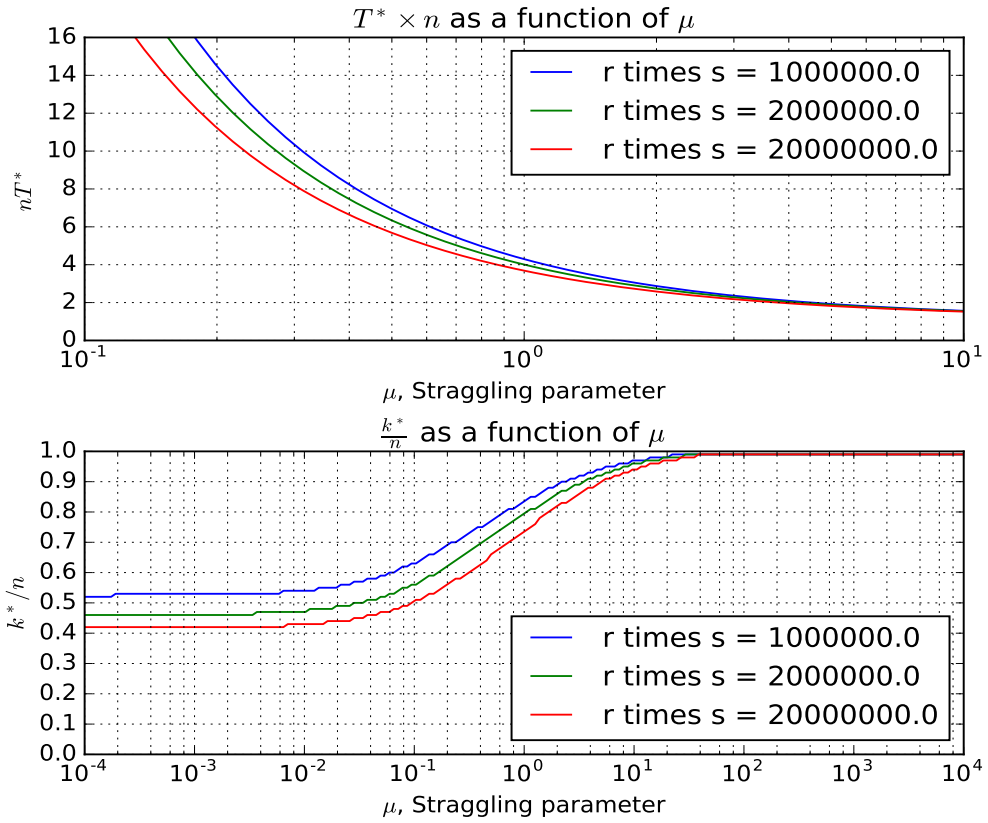


Figure 7.2: Parameters that minimize the overall runtime for different sizes of $\boldsymbol{A}$

In Fig 7.3 the straggling parameter is adapted to the size of the computation. The biggest matrix is taken as reference. This means, the straggling parameter results from the runtime distribution of one server performing the whole computation on the tallest matrix. This does not change the observed effect on the code rate. The more impact the decoding has, the higher is the optimal code rate. For the runtime on the other hand, a qualitative change in the behavior is observed. As expected, the taller the matrix, the more time the computation takes.

In conclusion, privacy constraints and decoding complexity favor higher code rates when the overall runtime should be minimized. While even slight privacy requirements influence the code rate gravely, the impact of the decoding complexity depends strongly on the proportion of the code length to the matrix size.

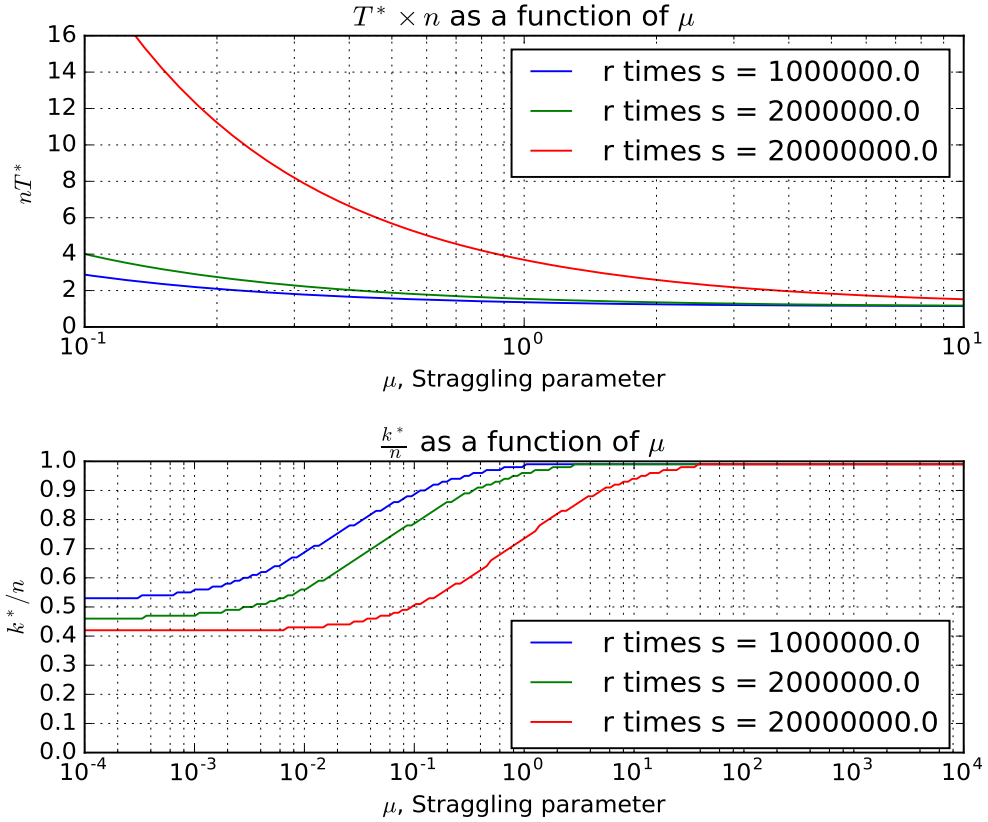


Figure 7.3: Optimal parameters with adjusted straggling parameters. The tallest matrix acts as reference.

8

Conclusion and Future Work

In this thesis a coding scheme was presented which guarantees privacy of data and mitigates the straggler effect in distributed computations. Already well studied structures from straggler mitigation were utilized in combination with random source symbols to shorten the computational delay and preserve privacy at the cost of additional computational load. The code parameters have been chosen to minimize the overall expected runtime, composed of the computational delay and decoding time, of the computation. It was observed that privacy favors higher code rates compared to standard, non-private computations because an increased code rate decreases the additional privacy related computational load.

The approach in this thesis is based on MDS codes which have fixed code dimensions. An alternative idea is to use rateless codes. Instead of defining code dimensions a priori without knowledge of the actual straggler behavior of the network, code symbols are generated on demand. Due to this adaptive procedure, unnecessary aggregation of resources is omitted. Recently, this approach gained more and more attention in the light of distributed computing. Because of the random encoding of rateless codes we tried to assure privacy by keeping the encoding secret instead of generating random source symbols which lead to an increased computational load. Unfortunately, we were not able to come up with a scheme where we can proof privacy. But the shown potential of these codes to mitigate the effect of stragglers encourages more intense research into private, distributed computing schemes based on rateless codes.

Bibliography

[1] L. A. Barroso, J. Clidaras, and U. Hoelzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool, 2013.

[2] "Google Cloud Platform." [Online]. Available: https://cloud.google.com/

[3] "Microsoft Azure." [Online]. Available: https://azure.microsoft.com/

[4] "Amazon Web Services." [Online]. Available: https://aws.amazon.com/

[5] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, pp. 74–80, 2013.

[6] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler Mitigation in Distributed Optimization Through Data Encoding," *31st Conference on Neural Information Processing Systems*, 2017.

[7] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2018, pp. 857–866.

[8] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *2016 IEEE Globecom Workshops (GC Wkshps)*, Dec 2016, pp. 1–6.

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

[10] A. Serverinson, "Coding for Distributed Computing," *Chalmers University of Technology*, 2017.

[11] A. Severinson, A. Graell i Amat, and E. Rosnes, "Block-diagonal coding for distributed computing with straggling servers," *2017 IEEE Information Theory Workshop (ITW)*, pp. 464–468, 2017.

[12] A. Severinson, A. G. i Amat, and E. Rosnes, "Block-diagonal and LT codes for distributed computing with straggling servers," *CoRR*, vol. abs/1712.08230, 2017.

[13] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[14] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[15] E. R. Berlekamp, *Algebraic coding theory [by] Elwyn R. Berlekamp*. McGraw-Hill New York, 1968.

[16] J. Massey, "Shift-register synthesis and bch decoding," *IEEE transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, 1969.

[17] G. Garrammone, "On decoding complexity of reed-solomon codes on the packet erasure channel," *IEEE Communications Letters*, vol. 17, no. 4, pp. 773–776, 2013.

[18] S. Kumar, E. Rosnes, and A. Graell i Amat, "Secure repairable fountain codes," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1491–1494, Aug 2016.

[19] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja, *A First Course in Order Statistics (Classics in Applied Mathematics)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008.

[20] R. Schlegel, 2018. [Online]. Available: https://github.com/ReentSchlegel/OptimalGroupingParameters.git