# Secure Control in the Cloud

## Using Multiparty Computation



Andrea Løvemærke


Master's Thesis

Control & Automation


AALBORG UNIVERSITY

DENMARK

Frontpage design: Each cloud has a specific colour corresponding to a coloured piece of the lock. The lock needs shares from each cloud to unlock and reconstruct the secret.
Fun fact: The numbers in the clouds are shares of first degree polynomials of the finite field with cardinality prime $p = 7979490791$. Every 10 digits constitute a share. If they are reconstructed in their respective order the results are 3 1 4 1 5 9 2 6 5 that are the first nine digits of pi.
Created in collaboration with Rasmus Gundorff Sæderup.

AALBORG UNIVERSITY
STUDENT REPORT

Master's Thesis
Control & Automation
Department of Electronic Systems

**Title:**
Secure Control in the Cloud
Using Multiparty Computation

**Project Period:**
February 2019 - June 2019

**Project Group:**
CA10 - 1037

**Student:**
Andrea Victoria Tram Løvemærke

**Supervisor:**
Rafael Wisniewski

**Pages:**
61

**Completion Date:**
June 14, 2019

**Abstract:**

The society of today and even more so in the future relies on data to reduce resource consumption, wastage and overall costs. Technologies have been developed such that devices across different functionalities can communicate and make independent decisions to optimize their individual function. Online control algorithms are needed to achieve this kind of linked optimization and automation. However, online servers are not guaranteed to be trustworthy, leading to great risk when data is released to third parties in exchange for computed control actions. It is vital that privacy is preserved and data is not compromised through acts of terror, to ensure safety in today's data-driven society.

This thesis is devoted to securing data within cloud computed control. Multiparty computation using secret sharing is investigated as a secure approach where private data is split into shares. Computing on shares can be performed on online servers with minimum risk as the shares individually do not disclose information about the private data. The data owner can reconstruct the private result of the secure multiparty computations with no other party gaining knowledge of the result.

It is investigated how unconstrained and equality constrained model predictive control problems can be securely solved in the cloud. The solution utilizes Gaussian elimination without pivoting and has proved to be useful. The thesis concludes that further research must be done within this field prior to become suitable for real life control problems. It does, however, show tremendous potential to becoming epochal for future online control applications.

# Preface

This report constitutes the thesis of the Master's program Control & Automation at Aalborg University and is authored by Andrea Victoria Tram Løvemærke. The thesis covers 30 ETCS points and is developed from the 11$^{th}$ of February 2019 to the 14$^{th}$ of June 2019.

Source referencing complies with the Harvard method and a bibliography with full citation of the referenced literature is found at the end of the report. Graphs and figures without a source reference are originals produced in MATLAB and Lucidchart, respectively.
All code used for simulations and implementation can be accessed at the GitHub repository https://github.com/avtl/thesis. The simulation results and the MATLAB code used to evaluate the results can moreover be found at this location.

Special gratitude and acknowledgement to my supervisor Rafael Wisniewski for his extraordinary and highly skilled guidance in this thesis. A great thank you goes to Katrine Sofie Tjell for her competent assistance and help when needed.
Moreover Rasmus Gundorff Sæderup deserves recognition for his valued contribution in discussing and solving everyday problems.

Lastly I must express my sincere appreciation to Aalborg University for my education in its entirety.

Andrea Victoria Tram Løvemærke,
Aalborg University, June 2019

# Danish Abstract

I nutidens samfund ønsker vi at få det maksimale ud af vores tid, vores penge og ressourcer i det hele taget. Således er flere teknologier udviklet med netop denne ambition - vores telefoner er blevet smarte, vores hjem er blevet delvist smarte og vores biler bliver i højere grad smarte. Ultimativt vil disse avanceringer udgøre et smart samfund, hvor alle enheder på tværs af funktionalitet kommunikerer og tager selvstændige beslutninger for at optimere deres individuelle funktion. Online reguleringsalgoritmer er nødvendige for at opnå denne form for sammenkædet optimering og automatisering. Online servere kan ikke garanteres at være troværdige, hvorfor det er risikabelt, at data i øjeblikket frigives til disse tredjeparter i bytte for beregning af reguleringsinput. Data er værdifuldt og af endnu større værdi hvis dataet betragtes i helhed. Det bør derfor hemmeligholdes for tredjeparter for at sikre brugerne mod overvågning, manipulation og i værste fald personskade, hvis ondsindede aktører ønsker at udvirke terror.
Denne specialeafhandling undersøger, hvorledes reguleringsalgoritmer kan beregnes i 'skyen' uden at data kan aflures eller inficeres.

Først præsenteres indledende teori om sikker distribueret beregning og en *secret sharing* metode. Herved kan data splittes op i dele på vilkårlig vis, så delene enkeltvis ikke har en værdi. Beregninger på disse datadele kan udføres på online servere uden risiko for at dataet misbruges af tredjeparter, da datadelene hver især ikke afslører information om det hemmelige data. Datadelene for beregningsresultatet kan rekonstrueres i den enkelte enhed, hvorfor denne vil være den eneste, der har viden om sit system og reguleringsinput.
Derefter undersøges det, hvordan model prædiktiv reguleringsproblemer kan beregnes sikkert i 'skyen'. Løsningsforslaget anvender Gaussisk eliminering uden pivotering og har vist sig brugbar. Det skal dog bemærkes, at en bedre måde for sikker invertering bør udvikles førend det fulde potentiale for løsningsforslaget kan udvindes. Parametre der har indflydelse på dette præcisionstab er karakteriseret og refleksioner i relation til reguleringsformål er siden beskrevet. Derudover betragtes processeringsomkostningerne for løsningsforslaget i forhold til anvendeligheden inden for realtids regulering. Det har vist sig urealiserbart krævende i forhold til traditionelle beregninger, hvorfor optimering af løsningsforslagets procceseringsomkostninger bør undersøges.

Distribuerede beregninger og hemmeligholdelse af data er et forskningsområde der i stigende grad bliver uundværligt, som den teknologiske udvikling, der baseres på private data, fortsætter. Arbejdet i denne specialeafhandling er initielt og bør fortsættes.

# Notation

| | |
|---|---|
| $:=$ | Definition |
| $\mathbf{I}_n$ | $n \times n$ identity matrix |
| $\mathbf{0}^{m \times n}$ | $m \times n$ zero-matrix |
| $\overset{?}{\neq}$ | Equality test: returns 1 when not equal |
| $\overset{?}{=}$ | Equality test: returns 1 when equal |
| $\text{diag}(\mathbf{v})$ | Diagonal matrix $\in \mathbb{K}^n$ where $\mathbf{v} = (v_1, \ldots, v_n)$ are the diagonal entries |
| $a$ | Public (open) value |
| $[\![a]\!]$ | Private value |
| $\mathbf{A}$ | Matrix |
| $\mathbf{a}$ | Vector |
| $\wedge$ | *AND* operator |
| $\vee$ | *OR* operator |
| $\hat{u}(k+i\|k)$ | $\hat{u}$ at time sample $k+i$ evaluated at time sample $k$ |
| $*$ | Schur product (entrywise multiplication) |
| $\det \mathbf{A}_{n-k}$ | Leading principal minor of order $k$ |
| $\mathbf{A}\|\mathbf{b}$ | Augmented system |

# Abbreviations

| | |
|---|---|
| *SECURE* | Secure Estimation and Control Using Recursion and Encryption |
| $\text{MPC}_{\text{omp}}$ | Multiparty Computation |
| $\text{MPC}_{\text{ontrol}}$ | Model Predictive Control |
| SSS | Secret Sharing Scheme |
| GE | Gaussian Elimination |
| RP | Raspberry Pi |
| TCP | Transmission Control Protocol |

# Nomenclature

**Chapter 2**

| | |
|---|---|
| $n$ | Number of parties |
| $P_1, \ldots, P_n$ | Parties |
| $[\![x_1, \ldots, x_n]\!]$ | Private information |
| $y$ | Function output |
| $\mathbb{F}_p$ | Finite field with cardinality prime $p$ |
| $t$ | Polynomial degree |
| $[\![a_0, \ldots, a_t]\!]$ | Polynomial coefficients |
| $[\![s_1, \ldots, s_n]\!]$ | Shares of secret $s$ |
| $f(x)$ | Polynomial over $\mathbb{F}_p$ of degree $t$ |
| $C$ | Set of $n$ indices of shares |
| $\delta_i(x)$ | Lagrange basis polynomial |
| $\mathbf{r}$ | Recombination vector |

**Chapter 3**

| | |
|---|---|
| $A$ | State matrix |
| $B$ | Input matrix |
| $C$ | Output matrix |
| $x(k)$ | State vector |
| $u(k)$ | Control input vector |
| $y(k)$ | Measured output vector |
| $z(k)$ | Controlled output |
| $V(k)$ | Cost function |
| $r(k)$ | Reference signal |
| $H_p$ | Prediction horizon |
| $H_w$ | Window parameter |
| $H_u$ | Control horizon |
| $Q(i)$ | State penalty |
| $R(i)$ | Control input penalty |

| | |
|---|---|
| $\Delta\mathcal{U}(k)$ | Reformulated change in control input |
| $\mathcal{Q}$ | Reformulated state penalty |
| $\mathcal{R}$ | Reformulated control input penalty |
| $\Theta$ | Based upon system matrices |
| $\mathcal{E}(k)$ | Tracking error |
| $\mathcal{G}(k)$ | Defined as: $2\Theta^T\mathcal{Q}\mathcal{E}(k)$ |
| $\mathcal{H}$ | Defined as: $\Theta^T\mathcal{Q}\Theta + \mathcal{R}$ |
| $S_\mathcal{Q}$ | Square-root matrix of $\mathcal{Q}$ |
| $S_\mathcal{R}$ | Square-root matrix of $\mathcal{R}$ |

**Chapter 4**

| | |
|---|---|
| $[\![\mathbf{A}]\!]$ | Private input matrix |
| $[\![\mathbf{x}]\!]$ | Private solution to linear system |
| $[\![\mathbf{b}]\!]$ | Private observation vector |
| $[\![\mathbf{A}']\!]$ | Modified input matrix |
| $[\![\mathbf{b}']\!]$ | Modified observation vector |
| $\mathbf{U}$ | Upper Toeplitz matrix |
| $\mathbf{L}$ | Lower Toeplitz matrix |
| $[\![h]\!]$ | Private input value |
| $[\![t]\!]$ | Private input value |

**Chapter 5**

| | |
|---|---|
| $[\![d]\!]$ | Private random variable used in semi-secure equality test |
| $\mathbb{Q}$ | Field of rational numbers |
| $[\![z]\!]$ | Private random variable used in secure inversion |
| $[\![s_s^{-1}]\!]$ | Inverted scaled secret |
| $w$ | Product of secret and private random variable |
| $c_s$ | Scaling factor |
| $k$ | Inverted scaled and rounded variable |

**Chapter 7**

| | |
|---|---|
| $e_1, e_2$ | Percentage error deviation |
| $\sigma$ | Standard deviation |

# Contents


**1 Introduction** **1**

**2 Preliminaries** **3**
2.1 Multiparty Computation . . . . . 3
2.2 Secret Sharing . . . . . 5
2.3 Focus of the Thesis . . . . . 13

**3 Control Problem** **15**
3.1 Model Predictive Control . . . . . 15
3.2 Unconstrained $\text{MPC}_{\text{ontrol}}$ . . . . . 16
3.3 Constrained $\text{MPC}_{\text{ontrol}}$ . . . . . 20

**4 Secure Solution** **23**
4.1 Secure Unconstrained Case . . . . . 23
4.2 Solving Linear Systems Securely . . . . . 25

**5 Simulation** **29**
5.1 Secure Equality Test . . . . . 29
5.2 Subtracting Shares from an Open Value . . . . . 30
5.3 Secure Inversion . . . . . 31

**6 Implementation** **35**
6.1 Network Setup . . . . . 35
6.2 Setup Presentation . . . . . 37

**7 Results** **39**
7.1 Precision . . . . . 39
7.2 Computational Cost . . . . . 48

**8 Conclusion** **51**

**9 Future Work** **53**

**Bibliography** **55**

**A Protocol 2.2 Example** **57**

**B Secure Matrix Multiplication Example** **61**

# 1 | Introduction

Aalborg University has a 2016-2021 strategy called *Knowledge for the World*. A part of this strategy is to promote innovative interdisciplinary collaboration. To achieve this six interdisciplinary research projects have been selected by the AAU Executive Board. Each project aims to determine how to solve a large societal challenge [AAU Strategy, 2018].
This thesis is part of *SECURE* that is one of the interdisciplinary research projects. *SECURE* is an abbreviation for Secure Estimation and Control Using Recursion and Encryption. The interdisciplinary research project is lead by Rafael Wisniewski and integrates another 11 researches from mathematics, signal analysis, automation and control engineering and techno-anthropology [AAU TANT, 2018].

The focus of *SECURE* is to develop methods to guarantee the correctness of operational decisions in a data-driven society. Moreover it is essential to ensure the privacy of the parties involved in the operations [AAU Strategy, 2017].

We live in a modern data-driven society where everything must go faster, smarter, and always be as convenient for the users as possible. The users' needs must be covered immediately as they occur. To do this information about the user is necessary for different systems. Information that is considered private to third parties is of significant value if leaked. This information could for instance be ones political orientation, economic status, health situation, location etc. Moreover it shall be noted that if the information is combined and considered in its entirety, additional information of great value can be obtained.

A data-driven society requires a large number of parties to collaborate and share information among each other. These parties are often unknown to the clients and cannot be guaranteed to be trustworthy.

*SECURE* aims to tackle the societal challenge of reducing resource consumption, wastage and overall costs. This requires optimization based on information about every individual within the society. This constitutes a large risk to the society as well as the individual citizens. It is therefore essential to develop methods that allow secure computations on private data in order to minimize resource consumption without the risk.
The motivation of the thesis is secure control computations in the cloud for utilization of autonomous public transportation.

To achieve fully autonomous public transportation, information on each vehicle such as location, speed, end-destination, if it is occupied or not, if it needs to be refuelled/recharged etc. is necessary. This information is gathered in the cloud, where complex control algorithms are processed. The results of these computations must

be delivered as control inputs to each vehicle in real time. The complexity exists in the number of vehicles, the different paths they are to take depending on the users' desired end-locations and the uncontrollable obstacles being humans on foot or bike in the traffic. The control algorithms must ensure a comfortable ride for the passengers in an effective manner, where no collisions occur for any of the vehicles.

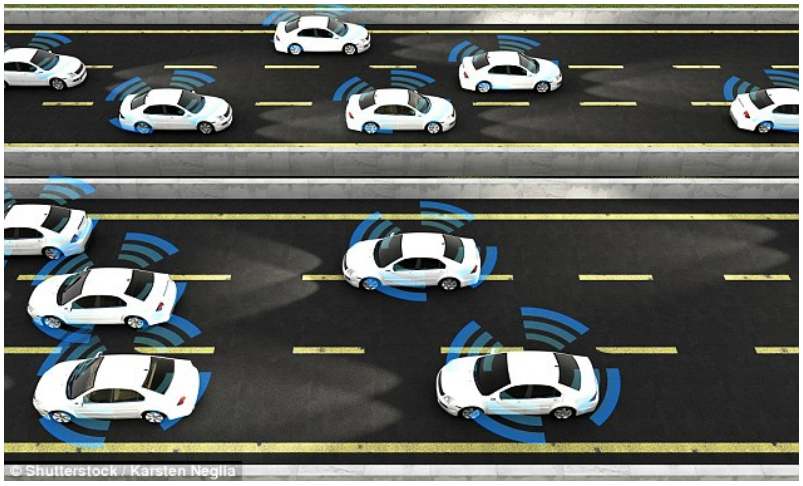


**Figure 1.1:** Conceptual image of identical autonomous vehicles securely controlled from the cloud [Hull, 2017].

The complexity and workload constitute a non-trivial challenge to solve. Moreover handling this challenging task must be done securely. If a society relies on an autonomous public transportation system it is crucial to keep it safe. If unauthorised parties obtain information it can be abused. It can reveal a user's daily routines, where the user works, how often the user visits the doctor, where friends and family live etc. Collecting this information can constitute great value to parties with interests that the user may not want to become subject to. Another important aspect to consider is how to avoid the control algorithms to be compromised as an act of terror to inflict traffic accidents.

A way to avoid this is by not pursuing this transportation advancement and remain with manually driven vehicles as of today. However, this is not reasonable nor satisfying for our modern society, hence this thesis aspires to contribute to applying secure control algorithms in the cloud as a partial solution to this challenging task. In the following chapter preliminaries are introduced to provide an understanding of what is meant by secure cloud computing.

# 2 | Preliminaries

Notice that [Tjell, 2018] is based on [Cramer et al., 2015]. As the thesis adopts the notation of [Tjell, 2018] it is chosen, where suited, to refer to this work instead of [Cramer et al., 2015].

This chapter introduces concepts that are fundamental in solving the challenges of secure computing in the cloud. It will later become obvious that the concepts in this chapter are inevitable and must be known to the reader before the thesis research can be comprehended. Note, that proofs of the utilized protocols will not be presented as prior work of *SECURE* has ensured correctness of the protocols [Tjell, 2018].

## 2.1 Multiparty Computation

Secure Multiparty Computation ($\text{MPC}_{\text{omp}}$) is based on a set of $n$ parties $P_1, ..., P_n$, that hold private information $[\![x_1, ..., x_n]\!]$, which must be combined in order to compute the function output $y = f(x_1, ..., x_n)$. The computation must be done without the parties learning each others inputs.
The function output must be correct and be the only new information released. One way to do this is by providing the parties' private inputs to a trustworthy third party, whom then computes the function output, see Figure 2.1. The third party can, however, not be guaranteed to be trustworthy, also known as *honest*, thus this is not a reliable way of obtaining the function output.

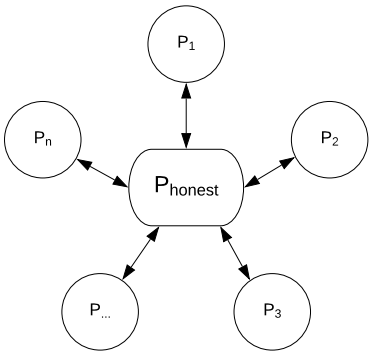


**Figure 2.1:** *Ideal world:* Parties sharing their private information with an honest third party to compute the result using secure communication channels.

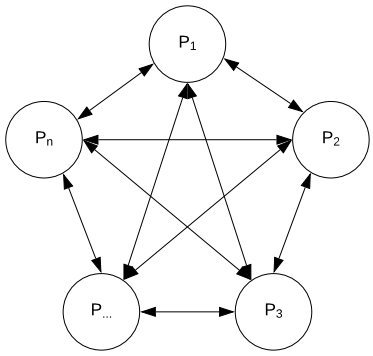


**Figure 2.2:** *Real world:* Parties distributing $n-1$ shares to each other to obtain the result without disclosing their private values using secure communication channels.

Secure $\text{MPC}_{\text{omp}}$ relies on sharing information between the parties, but without revealing their private value. The parties must cooperate to obtain the function out-

put, see Figure 2.2. Note, that all communication is assumed to be secure based on adequate encryption.

To preserve privacy secure protocols are defined and must be followed by all parties. It shall be noted, that this cannot always be assumed to be respected as corrupt parties may deviate from the protocol to obtain more information than intended or to cause an incorrect output. This is known as malicious behaviour and any entity behaving in such a way is called an adversary.
The adversary attacks a protocol by taking control of a subset of parties, which then become corrupt. An adversary can be either passive or active. A passive adversary is a simple adversary where all parties follow the protocol and only private information of the corrupted parties is disclosed, see Figure 2.3. An active adversary is a dynamic adversary where the corrupted subset of parties may arbitrarily deviate from the protocol aiming to manipulate the function output, see Figure 2.4. [Tjell, 2018, p. 6-7]

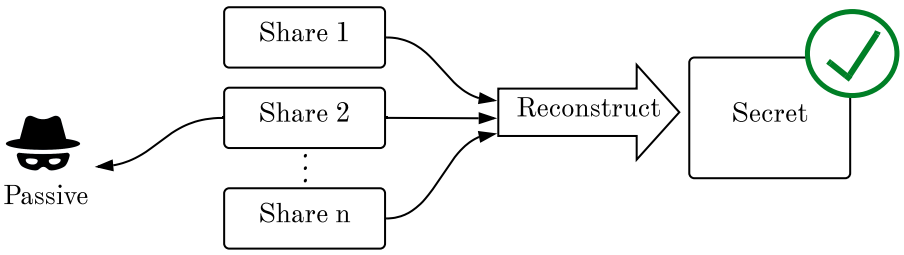


**Figure 2.3:** Passive adversary: Private information of corrupt parties is disclosed to the adversary but remains intact. Adversary object is from [Ahkâm, 2017].

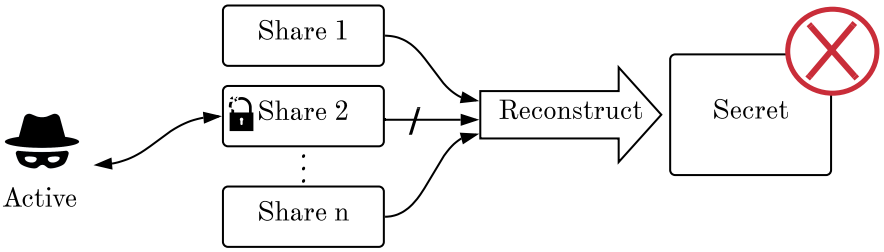


**Figure 2.4:** Active adversary: Private information of corrupt parties is disclosed and compromised in order to falsify the reconstructed function output. Adversary object is from [Ahkâm, 2017].

Different protocols exist to ensure security against malicious behaviour. The type of protocol determines what kind of security is obtained. Passive security can be obtained by simpler protocols compared to active security. Note, that an active secure protocol protects against both kinds of adversaries, as active adversaries manipulate inputs to falsify the output. This requires knowledge of the input information, which is what a passive adversary solely targets.

Before protocols to protect against adversaries can be defined, secure $\text{MPC}_{\text{omp}}$ requires the private values of the parties to be split into shares that individually has

no value. The shares are distributed among the parties, but without revealing the parties' private information. This is known as secret sharing and will be described in the following.

## 2.2 Secret Sharing

Secret sharing is a concept within cryptography that allows a secret to be distributed among multiple parties without revealing the secret to any of the parties. The secret is split into multiple shares according to a secret sharing scheme.
The Shamir's secret sharing scheme (Shamir's SSS) has been investigated in the *SECURE* research project and is deemed adequate for the intended use within this work.

### 2.2.1 Shamir's Secret Sharing Scheme

Shamir's SSS constructs $n$ shares of a secret $s \in \mathbb{F}_p$, using Lagrange polynomials of degree $t < n$ [Cramer et al., 2015, p. 33-35]. Note, that $p$ is a prime and defines the cardinality of a finite field. A polynomial $f(x) \in \mathbb{F}_p$ of degree $t < n$ is secure for $t$ corrupt parties as these will not be able to reconstruct the secret from their shares. The secret can, however, easily be reconstructed from shares of any $t + 1$ or more parties using Lagrange interpolation. It is therefore essential to consider what polynomial degree will be sufficient to secure data depending on the number of parties and the value and nature of the data being handled.

The polynomial is constructed such that $f(0)$ equals the secret and the remaining coefficients are random. The Shamir's SSS protocol is given as:

---

**Protocol 2.1** Shamir's secret sharing scheme [Tjell, 2018, p. 14]

---

**Input:** Let $s \in \mathbb{F}_p$ be the secret and the polynomial degree be $t < n$ where $n$ is the number of parties.
**Output:** The shares $[\![s_1,..., s_n]\!]$ of $s$.

1: Select random uniformly distributed polynomial coefficients $a_1, ..., a_n \in \mathbb{F}_p^t$.
2: Define the polynomial $f(x) = a_0 + a_1x + ... + a_tx^t \in \mathbb{F}_p$. Note that $a_0 = s = f(0)$.

3: Define the shares of $s$ as

$$\begin{aligned} s_1 &= f(1) \\ &\vdots \\ s_n &= f(n) \end{aligned}$$

---

Lagrange interpolation that is used to reconstruct the secret $s \in \mathbb{F}_p$ is described in the following.

### 2.2.2 Lagrange Interpolation

Having shares from $t+1$ or more parties it is possible to reconstruct the secret $s \in \mathbb{F}_p$. The polynomial of degree $t$ is constructed as [Cramer et al., 2015, p. 33-35]:

$$f(x) = \sum_{i \in C} s_i \delta_i(x) \ , \tag{2.1}$$

where:

| | |
|---|---|
| $f(x)$ | is the polynomial over $\mathbb{F}_p$ of degree $t$, |
| $C$ | is the set of $n$ indices of the shares, |
| $\delta_i(x)$ | is the Lagrange basis polynomial. |

Note, that the Lagrange basis polynomial is defined as:

$$\delta_i(x) = \prod_{j \in C, j \neq i} \frac{x-j}{i-j} \ . \tag{2.2}$$

Lagrange interpolation requires $t+1$ points of $f(x)$ to reconstruct the secret shared by Shamir's SSS of $t$-order Lagrange polynomials. The Lagrange basis polynomials equal either 0 or 1 depending on the indices evaluated upon [Tjell, 2018, p. 15]:

$$\delta_i(k) = \prod_{j \in C, j \neq i} \frac{k-j}{i-j} = \frac{k-1}{i-1} ... \frac{k-k}{i-j} ... \frac{k-(t+1)}{i-(t+1)} = 0 \text{ for } k \neq i \ , \tag{2.3}$$

$$\delta_i(i) = \prod_{j \in C, j \neq i} \frac{i-j}{i-j} = 1 \ . \tag{2.4}$$

Note, that it is not necessary to reconstruct the entire polynomial $f(x)$ as it is only the secret that is of interest to reconstruct. The secret is equal to $f(0)$ hence the Lagrange basis polynomial in Equation 2.2 can be simplified to:

$$r_i = \delta_i(0) = \prod_{j \in C, j \neq i} \frac{-j}{i-j} \ . \tag{2.5}$$

All simplified Lagrange basis polynomials are gathered in a recombination vector:

$$\mathbf{r} = [r_1, ..., r_n] \ . \tag{2.6}$$

The recombination vector, $\mathbf{r}$, is independent of $f(x)$ and can therefore be utilized with all polynomials of degree $t < n$. This allows Shamir's SSS, see Protocol 2.1, to be simplified to:

$$f(0) = \sum_{i \in C} r_i s_i \ . \tag{2.7}$$

The secret $s = f(0)$ hence Equation 2.7 is sufficient to compute in order to reconstruct the secret.
A more convenient notation is now introduced.

### 2.2.3 Notation and Protocols

To ease the reading of protocols to come, the simplest and most used notation in the research field is first presented. The thesis follows the notation of [Cramer et al., 2015, p. 37-38].

$[\![a; f]\!]_t$ holds the following information:

- $a \in \mathbb{F}_p$ is the secret
- $f \in \mathbb{F}_p$ is a random polynomial
- $t < n$ is the polynomial degree

Note, that the secret $a$ is split into $n$ shares calculated from $f(x)$ [Tjell, 2018, 20-21], such that:

$$[\![a; f]\!]_t = [\![f(1), ..., f(n)]\!] \ . \tag{2.8}$$

Polynomials can express any secret that is a finite function over a finite field $\mathbb{F}_p$ thus addition and multiplication are the only required operations as these can evaluate any polynomial [Tjell, 2018, p. 24]. This section presents addition and multiplication in secure $\text{MPC}_{\text{omp}}$ using Shamir's SSS.

**Operations using the notation:**
Given that $a,\ b,\ c \in \mathbb{F}_p$ and $f(x)$ and $g(x)$ are polynomials over $\mathbb{F}_p$.

$$[\![a; f]\!]_t + [\![b; g]\!]_t = [\![a + b; f + g]\!]_t \ , \tag{2.9}$$
$$c[\![a; f]\!]_t = [\![ca; cf]\!]_t \ , \tag{2.10}$$
$$[\![a; f]\!]_t * [\![b; g]\!]_t = [\![ab; fg]\!]_{2t} \ . \tag{2.11}$$

Note, that $*$ denotes the Schur product [Cramer et al., 2015, p. 14].

It is essential to observe, that the polynomial degree in Equation 2.11 is increased. The cause for this is illustrated by the derivations:

$$\begin{aligned} [\![a; f]\!]_t * [\![b; g]\!]_t &= [\![f(1), ..., f(n)]\!] * [\![g(1), ..., g(n)]\!] && (2.12)\\ &= [\![f(1)g(1), ..., f(n)g(n)]\!] && (2.13)\\ &= [\![(fg)(1), ..., (fg)(n)]\!] && (2.14) \end{aligned}$$

$$\begin{aligned} (fg)(x) &= (a + a_1x + ... + a_tx^t) \cdot (b + b_1x + ... + b_tx^t)\\ &= ab + (a_1b_2 + a_2b_1)x + ... + (a_tb_t)x^{2t} \ . && (2.15) \end{aligned}$$

Each term in Equation 2.14 can be expanded as seen in Equation 2.15.
The demonstrated behaviour of increasing polynomial degree is for multiplication of two private values only. The polynomial degree increases according to the number

of private values that are being multiplied, hence the computational challenge can quickly become immense.

Protocol 2.2 multiplies two private polynomials and outputs the product as a polynomial of degree $t$.

**Protocol 2.2** Multiplication [Tjell, 2018, p. 23]

**Input:** Let the parties hold $[\![a; f_a]\!]_t$, $[\![b; f_b]\!]_t$ and the degree be $t < \frac{n}{2}$.
**Output:** $[\![y; f_y]\!]_t$ where $y = ab$.

1: The parties compute

$$[\![ab; h]\!]_{2t} = [\![a; f_a]\!]_t * [\![b; f_b]\!]_t \ , \tag{2.16}$$

where $h = f_a f_b$ that is the $2t$ degree polynomial according to Equation 2.11. Note, that $h(0) = ab$.

2: The party $P_i$ distributes $[\![h(i); f_i]\!]_t$.

3: The parties compute

$$\sum_{i=1}^{n} r_i [\![h(i); f_i]\!]_t = [\![\sum_{i=1}^{n} r_i h(i); \sum_{i=1}^{n} r_i f_i]\!]_t \ , \tag{2.17}$$

$$= [\![h(0); \sum_{i=1}^{n} r_i f_i]\!]_t \ , \tag{2.18}$$

$$= [\![ab; \sum_{i=1}^{n} r_i f_i]\!]_t \ , \tag{2.19}$$

where $r_i$ is the $i^{\text{th}}$ entry of the recombination vector, see Equation 2.6.

Note, that $f_y = \sum_{i=1}^{n} r_i f_i$ hence all parties learn $[\![y; f_y]\!]$.

Party $P_i$ first multiplies its shares of $[\![a]\!]$ and $[\![b]\!]$. The polynomial $f_a f_b = h$ has the degree $2t$. Party $P_i$ then creates a new polynomial $f_i(x) = h(i) + c_1 x + \ldots + c_t x^t$, where $c$ is a random uniformly distributed variable. Note, that the new polynomial $f_i$ has degree $t$. $P_i$ distributes its shares of $f_i(x)$, such that the parties hold:

| | $f_1(x)$ | $f_2(x)$ | $f_3(x)$ |
|---|---|---|---|
| $P_1$ | $f_1(1)$ | $f_2(1)$ | $f_3(1)$ |
| $P_2$ | $f_1(2)$ | $f_2(2)$ | $f_3(2)$ |
| $P_3$ | $f_1(3)$ | $f_2(3)$ | $f_3(3)$ |

Party $P_i$ uses the recombination vector, see Equation 2.5 and Equation 2.6, to calculate the resulting shares of $ab$. The recombination vector is publicly known,

thus all parties can utilize it. The resulting shares are determined as follows:

$$P_1 : \ r_1 \cdot f_1(1) + r_2 \cdot f_2(1) + r_3 \cdot f_3(1) \ , \tag{2.20}$$
$$P_2 : \ r_1 \cdot f_1(2) + r_2 \cdot f_2(2) + r_3 \cdot f_3(2) \ , \tag{2.21}$$
$$P_3 : \ r_1 \cdot f_1(3) + r_2 \cdot f_2(3) + r_3 \cdot f_3(3) \ . \tag{2.22}$$

The resulting shares of $ab$ can now be used in future computations in $\text{MPC}_{\text{omp}}$ without the polynomial degree escalating as the polynomial degree remains $t$. To reconstruct the result of the multiplication, the resulting shares are multiplied by the recombination vector:

$$\begin{aligned} ab = & r_1(r_1 \cdot f_1(1) + r_2 \cdot f_2(1) + r_3 \cdot f_3(1)) && (2.23)\\ & + r_2(r_1 \cdot f_1(2) + r_2 \cdot f_2(2) + r_3 \cdot f_3(2)) && (2.24)\\ & + r_3(r_1 \cdot f_1(3) + r_2 \cdot f_2(3) + r_3 \cdot f_3(3)) \ . && (2.25) \end{aligned}$$

An example of Protocol 2.2 is calculated in Appendix A.

Protocol 2.2 is an inefficient multiplication protocol as it relies on distributing shares between parties, in order to obtain the result. Distributing shares in a real network require encrypted communication that can be time consuming. Computation time affects the performance of real time control negatively, thus it is vital to reduce it as much as possible.
A more efficient method is the *Beaver's triplet* multiplication protocol. It relies on three random variables, which are $[\![\alpha; f_\alpha]\!]_t$, $[\![\beta; f_\beta]\!]_t$ and $[\![\gamma; f_\gamma]\!]_t$, where the *Beaver's triplet* is $\gamma = \alpha\beta$. The shares $\alpha$, $\beta \in \mathbb{F}_p$ are random uniformly distributed and unknown to all parties. A *Beaver's triplet* is created using the *Beaver's trick*:

**Protocol 2.3** *Beaver's trick* [Tjell, 2018, p. 25-26]

**Input:** Let the polynomial degree be $t < \frac{n}{2}$.
**Output:** $[\![\alpha; f_\alpha]\!]_t$ , $[\![\beta; f_\beta]\!]_t$ , $[\![\gamma; f_\gamma]\!]_t$.

1: The shares $[\![\alpha_i; f_{\alpha_i}]\!]_t$ and $[\![\beta_i; f_{\beta_i}]\!]_t$ must be distributed by the party $P_i$ for $i = 1, ..., n$.
Note, that $\alpha_i, \beta_i \in \mathbb{F}_p$ must by random uniformly distributed.

2: Party $P_i$ computes

$$[\![\alpha_; f_\alpha]\!]_t = \sum_{i=1}^{n} [\![\alpha_i; f_{\alpha_i}]\!]_t \ , \tag{2.26}$$
$$[\![\beta_; f_\beta]\!]_t = \sum_{i=1}^{n} [\![\beta_i; f_{\beta_i}]\!]_t \ . \tag{2.27}$$

3: Lastly all parties compute

$$[\![\gamma; f_\gamma]\!]_t = [\![\alpha; f_\alpha]\!]_t * [\![\beta; f_\beta]\!]_t \ , \tag{2.28}$$

using Protocol 2.2.

Note, that the *Beaver's trick* in Protocol 2.3 invokes the time consuming multiplication protocol, Protocol 2.2, to create *Beaver's triplets*, thus it is essential that it is performed as a preprocessing phase. In the case of autonomous public transportation, creating *Beaver's triplets* can for instance be done while the vehicles are parked and thus not relying on efficient real time control computations. A *Beaver's triplet* can only be used for one multiplication operation thus it is necessary to create sufficiently many *triplets* to preserve privacy [Tjell, 2018, p. 26].

The *Beaver's triplet* multiplication protocol is more efficient than Protocol 2.2, given that sufficiently many *triplets* are available, as it broadcasts information that has been changed by a *triplet*. Note, that broadcasting a value is defined by letting all parties know the value, thus encrypted communication is not required.
This allows the parties to compute the function output without distributing shares among each other. The private product is ultimately obtained by addition. This will become clear from Protocol 2.4 and supplemental derivations. The *Beaver's triplet* multiplication protocol is presented as follows.

**Protocol 2.4** *Beaver's triplet* for Multiplication [Tjell, 2018, p. 26]

**Input:** Let the parties hold $[\![a; f_a]\!]_t$, $[\![b; f_b]\!]_t$, $[\![\alpha; f_\alpha]\!]_t$, $[\![\beta; f_\beta]\!]_t$, $[\![\gamma; f_\gamma]\!]_t$, where the latter three are the *Beaver's triplet*.
**Output:** $[\![y; f_y]\!]_t$, where $y = ab$.

1: The parties compute

$$[\![d; f_d]\!]_t = [\![a; f_a]\!]_t - [\![\alpha; f_\alpha]\!]_t \ , \tag{2.29}$$
$$[\![e; f_e]\!]_t = [\![b; f_b]\!]_t - [\![\beta; f_\beta]\!]_t \ . \tag{2.30}$$

2: The parties broadcast $d$ and $e$.

3: The parties compute

$$[\![y; f_y]\!]_t = de + d\,[\![\beta; f_\beta]\!]_t + [\![\alpha; f_\alpha]\!]_t\, e + [\![\gamma; f_\gamma]\!]_t \ . \tag{2.31}$$

All parties learn the function output as each party can compute $y = ab$ from the broadcast values $d$ and $e$ according to Equation 2.31. As the coefficients of the *Beaver's triplet* are random uniformly distributed, thus are $d$ and $e$, and the privacy of $a$ and $b$ is therefore not compromised.
To illustrate that Equation 2.31 returns the product of $a$ and $b$, the following derivations are presented.

From Equation 2.29-2.30 it is seen that:

$$d = [\![a]\!] - [\![\alpha]\!] \qquad [\![\alpha]\!] = [\![a]\!] - d \qquad [\![a]\!] = d + [\![\alpha]\!] \tag{2.32}$$
$$e = [\![b]\!] - [\![\beta]\!] \qquad [\![\beta]\!] = [\![b]\!] - e \qquad [\![b]\!] = e + [\![\beta]\!] \tag{2.33}$$

Hence the product $y = ab$ can be expressed as:

$$\begin{aligned} y = ab &= (d + [\![\alpha]\!])(e + [\![\beta]\!]) && (2.34)\\ &= de + d\,[\![\beta]\!] + [\![\alpha]\!]\,e + [\![\alpha]\!][\![\beta]\!] && (2.35)\\ &= de + d\,[\![\beta]\!] + [\![\alpha]\!]\,e + [\![\gamma]\!] && (2.36) \end{aligned}$$

Equation 2.36 is identical to Equation 2.31, hence the *Beaver's triplet* for multiplication has proved not to increase the polynomial degree.

This concludes the description of multiplication of shares on polynomial form. In the following secure addition is presented.

Addition is a simpler operation, where each party adds its shares and exchanges the result. This also applies for Shamir's SSS polynomials, as the corresponding coefficients in the shared polynomials are simply added with no affect on the polynomial degree. A protocol for this is provided in the following.

---

**Protocol 2.5** Addition [Tjell, 2018, p. 22]

---

**Input:** Let the parties hold $[\![a; f_a]\!]_t$, $[\![b; f_b]\!]_t$ and the degree be $t < n$.
**Output:** $[\![y; f_y]\!]_t$ where $y = a+b$.

1: The parties compute

$$[\![y; f_y]\!]_t = [\![a; f_a]\!]_t + [\![b; f_b]\!]_t \; . \tag{2.37}$$

Note, that $f_y = f_a + f_b$.

---

This concludes the necessary operations for polynomial evaluation in a finite field. It shall be noted, that the protocols presented are only secure for passive adversaries as corrupt parties can compromise the shares and thus manipulate the output. According to [Tjell, 2018, section 2.4] different approaches to obtain active security can be utilized as an addition to the presented protocols in this chapter. The focus of the thesis must be delimited, thus it does not consider protection against active adversaries as it has already been investigated within the *SECURE* project.

Shamir's SSS and the use of additive polynomial evaluation is exemplified in the following to improve the reader's understanding of the concepts.

### 2.2.4 Example

Three parties $P_1$, $P_2$ and $P_3$ are voting *approve* or *not approve* to a decision equivalent to 1 or 0, respectively. If the sum is greater than half the number of parties, the decision is approved. As there are three parties there must be at least two parties voting *approve* for the decision to be approved. In this example $P_1$ and $P_2$ vote *approve* and $P_3$ votes *not approve*.
Using secure $\text{MPC}_{\text{omp}}$ and Shamir's SSS the parties $P_1$ and $P_2$ will not know that $P_3$ voted *not approve*, hence keeping the vote anonymous.

Shamir's SSS computes shares based on Lagrange polynomials of degree $t$. Summing these polynomials equals a new polynomial of the same degree $t$.

In this example the polynomial degree is $t = 2$ as $t < n$.
The polynomials can for instance be:

$$f_1(x) = 1 - 1.5\ x + 0.5\ x^2\ , \tag{2.38}$$
$$f_2(x) = 1 - 3\ x + x^2\ , \tag{2.39}$$
$$f_3(x) = 0 + 1.5\ x - 0.5\ x^2\ . \tag{2.40}$$

Note, that $f(x) = f_1(x) + f_2(x) + f_3(x)$ and that $f(0) = 2$ is the secret.
The coefficients in Equation 2.38-2.40 are random uniformly distributed except from the secrets. It is defined that the polynomials must go through the parties' secrets being either 1 or 0 according to their voting. The three polynomials, Equation 2.38-2.40, are plotted in Figure 2.5.

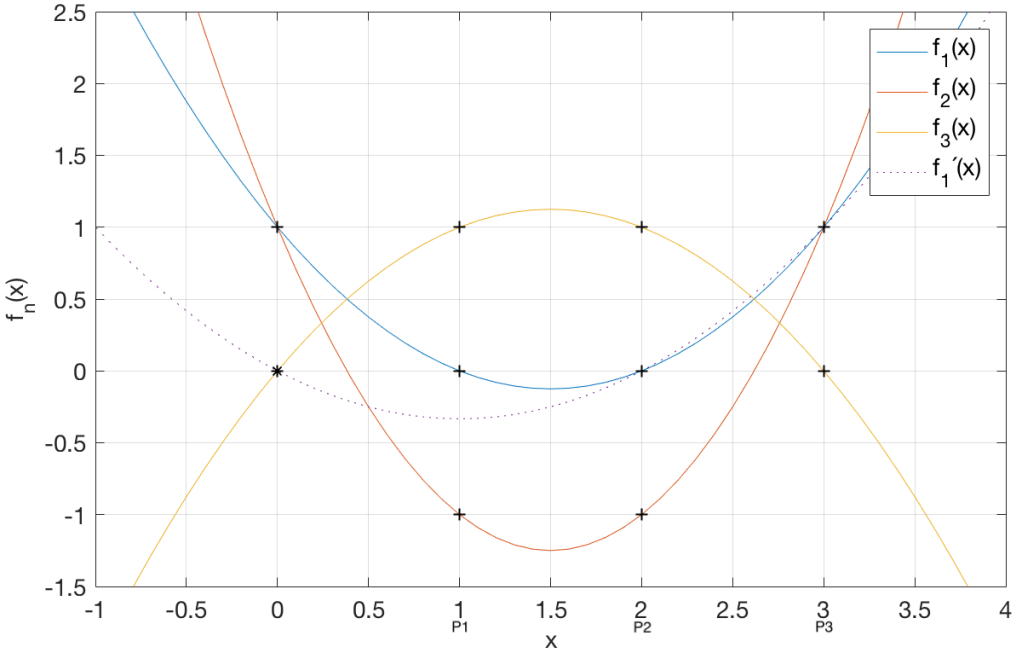


**Figure 2.5:** Equation 2.38-2.40 plotted as an example to illustrate how $t+1$ shares are necessary to reconstruct the secret $s \in \mathbb{F}_p$. Note, that the dotted polynomial, $f_1'(x)$, is an example of how only $t$ corrupt parties can mistakenly conclude a false answer from $P_1$.

Each party privately distributes one share to each of the other parties, such that:
$P_1$ holds the shares $f_1(1), f_2(1)$ and $f_3(1)$ allowing it to compute $f(1)$,
$P_2$ holds the shares $f_1(2), f_2(2)$ and $f_3(2)$ allowing it to compute $f(2)$,
$P_3$ holds the shares $f_1(3), f_2(3)$ and $f_3(3)$ allowing it to compute $f(3)$.

This yields:

$$f(1) = f_1(1) + f_2(1) + f_3(1) = 0 \,, \tag{2.41}$$
$$f(2) = f_1(2) + f_2(2) + f_3(2) = 0 \,, \tag{2.42}$$
$$f(3) = f_1(3) + f_2(3) + f_3(3) = 2 \,. \tag{2.43}$$

Note, that it is necessary to have $t+1$ shares to be able to reconstruct the secret. If less than $t+1$ parties collude, for example $P_2$ and $P_3$ sharing their information, the vote of $P_1$ cannot be disclosed. This is due to the existence of various second order polynomials that satisfy both the shares of $P_2$ and $P_3$ and moreover either $f(1) = 0$ or $f(1) = 1$. This prevents $t$ or less corrupt parties to disclose the vote of $P_1$. This is illustrated in Figure 2.5 by $f_1'(x)$ that is a polynomial, which satisfies the shares of $P_2$ and $P_3$, but with an incorrect answer of $P_1$'s vote.

Based on Equation 2.41-2.43 the polynomial that reconstructs the secret can be determined using Lagrange interpolation, see Equation 2.1 and Equation 2.2, as follows:

$$0 \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} + 0 \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} + 2 \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} = 2 - 3x + x^2 \tag{2.44}$$

The result of the Lagrange interpolation must equal the polynomial equivalent to the sum of the parties' individual polynomials, Equation 2.38-2.40, as:

$$f(x) = f_1(x) + f_2(x) + f_3(x) = 2 - 3x + x^2 \,. \tag{2.45}$$

It can be seen that the Lagrange interpolation in Equation 2.44 indeed yields the same polynomial as the summed polynomial shares, see Equation 2.45.

From the reconstructed polynomial in either Equation 2.44 or Equation 2.45 it is clear that the result of the vote is $f(0) = 2$, meaning the decision is approved. [SimplyScience, 2018]

Based on the preliminary theory on multiparty computation and secret sharing it is possible to specify the focus of the thesis.

## 2.3 Focus of the Thesis

The use of the existing protocols within $\text{MPC}_{\text{omp}}$ and secret sharing schemes cannot fulfil the demands of a solution for an autonomous public transportation system as there are too many parties (vehicles) to handle. To limit the number of parties

within secure $\text{MPC}_{\text{omp}}$ each vehicle must split its data into a limited number of shares and distribute each share to a cloud server. It decreases the communication significantly as $\text{MPC}_{\text{omp}}$ and secret sharing usually require the vehicle to distribute its shares to $n - 1$ parties, where $n$ for autonomous public transportation will be tens of thousands of vehicles.
Each cloud server will instead receive one share from each vehicle. The shares individually reveal nothing about a vehicles' private data, thus the cloud servers do not obtain any valued information. This is vital in order to preserve privacy and moreover the security of the autonomous public transportation system.

This work investigates how control can be securely computed in the cloud using multiparty computation.
The constellation of parties is delimited to a single vehicle, hereafter referred to as the data owner, and three cloud servers. The constellation is illustrated in Figure 2.6.

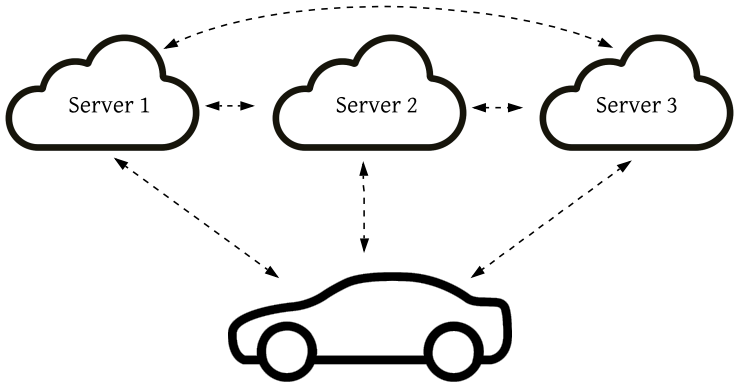


**Figure 2.6:** Three cloud servers and one data owner, being the autonomous vehicle. The dotted lines between the parties represent secure communication channels.

The constellation of parties can be expanded to include multiple vehicles, if the thesis determines an attainable approach for secure control computations in the cloud using $\text{MPC}_{\text{omp}}$.

The investigated control problem is described in the following chapter.

# 3 | Control Problem

Autonomous public transportation is a non-trivial control challenge even without the considerations of privacy. The thesis investigates how secure control can be facilitated in the cloud using $\text{MPC}_{\text{omp}}$. The control strategy itself is not the primary focus of the research, thus different control strategies can be examined.

It is chosen to investigate how Model Predictive Control ($\text{MPC}_{\text{ontrol}}$) can be computed securely as it is a widely used control strategy that handles constrained multivariable control problems naturally [Maciejowski, 2000, p. 1].
Two $\text{MPC}_{\text{ontrol}}$ cases are considered to establish the potentials within secure $\text{MPC}_{\text{omp}}$. The presented control theory serves the purpose of demonstrating how the two $\text{MPC}_{\text{ontrol}}$ cases can be expressed in a form that can be securely solved using $\text{MPC}_{\text{omp}}$. Control dynamics and details in general will therefore not be considered in depth.

Note, that the predefined bold type notation of vectors and matrices is neglected in this chapter. The chapter presents many lengthy equations of vectors and matrices that will seem overwhelming, if they are to respect the general notation.

## 3.1 Model Predictive Control

$\text{MPC}_{\text{ontrol}}$ optimizes a cost function $V(k)$ over a prediction horizon $H_p$ that yields a sequence of predicted optimal control inputs $\hat{u}(k)$, ... , $\hat{u}(k+i)$. Only the first sample of the predicted optimal control inputs is applied. A sample time later the procedure is repeated. This is known as receding horizon [Maciejowski, 2000, p. 7-9]. Figure 3.1 illustrates the principles of this.

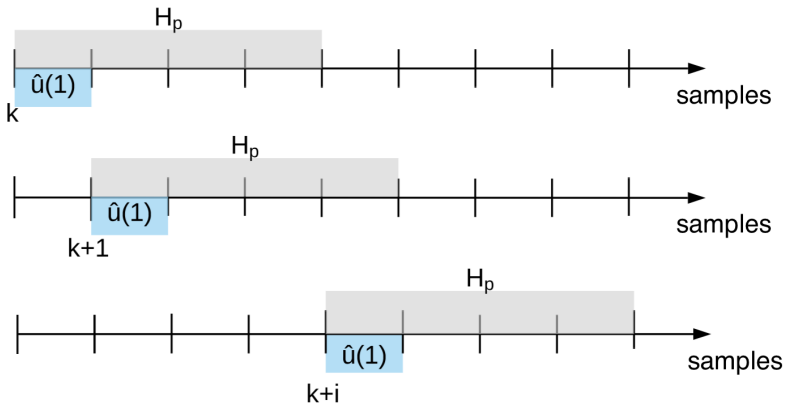


**Figure 3.1:** Receding horizon principles illustrated for three time samples, $k$, $k+1$ and $k+i$.

As the thesis is limited in resources, the research strategy is to investigate the simplest cases as a basis for future work. Further research must be conducted as a continuation of this work before secure control can be realised in real life.

An unconstrained and a constrained $\text{MPC}_{\text{ontrol}}$ problem, both with all states measured, will pose as the control problems of interest. In systems, where not all states are measured or great amount of noise is present, an observer can advantageously be implemented. Regarding the derivations to come the measured state vector can simply be replaced by the estimated state vector if an observer is implemented.

Firstly the unconstrained case is presented.

## 3.2 Unconstrained $\text{MPC}_{\text{ontrol}}$

This section is based on [Maciejowski, 2000, p. 74-79].

Many systems constitute Multiple Input Multiple Output(MIMO) systems that conveniently can be formulated on state space form, as:

$$x(k+1) = A\,x(k) + B\,u(k)\ , \tag{3.1}$$
$$y(k) = C_y\,x(k)\ , \tag{3.2}$$
$$z(k) = C_z\,x(k)\ , \tag{3.3}$$

where:

| | |
|---|---|
| $A$ | is the state matrix, |
| $B$ | is the system input matrix, |
| $C$ | is the system output matrix, |
| $x$ | is the system state vector, |
| $u$ | is the system input vector, |
| $y$ | is the measured output vector, |
| $z$ | are the controlled output vector. |

The cost function in $\text{MPC}_{\text{ontrol}}$ is a quadratic function that includes weight matrices to penalize signals to comply with desired performance dynamics. Quadratic functions are often expressed as *quadratic forms* where $x^TQx$ is compactly represented as $||x||^2_Q$. Note that $x$ must be a vector and $Q$ must be a symmetric matrix.

The cost function is defined as:

$$V(k) = \sum_{i=H_w}^{H_p-1} \left\| \hat{z}(k+i|k) - r(k+i|k) \right\|_{Q(i)}^2 + \sum_{i=0}^{H_u-1} \left\| \Delta\hat{u}(k+i|k) \right\|_{R(i)}^2 , \tag{3.4}$$

where:

| | |
|---|---|
| $r$ | is the reference, |
| $H_p$ | is the prediction horizon, |
| $H_w$ | is the window parameter, |
| $H_u$ | is the control horizon, |
| $Q(i)$ | is the positive semi definite output penalty matrix, |
| $R(i)$ | is the positive semi definite input penalty matrix. |

Note, that $\Delta\hat{u}$ is the change between the predicted future input $\hat{u}(k+i|k)$ and the current input $u(k)$, defined as:

$$\Delta\hat{u}(k+i|k) = \hat{u}(k+i|k) - u(k) . \tag{3.5}$$

The weighting matrices $Q(i)$ and $R(i)$ penalize the states and the change in control inputs, respectively, according to a desired performance. The sizes of the horizons $H_p$, $H_u$ and $H_w$ and the weighting matrices $Q(i)$ and $R(i)$ are considered tuning parameters. Choosing the sizes of the horizons must be done, such that:

$$\begin{aligned} &H_p > H_u, \\ &H_u \geq \text{slowest system dynamics}, \\ &H_w \geq 1. \end{aligned}$$

Equation 3.4 can be reformulated such that $\mathcal{U}$ and $\mathcal{Z}$ are expressed in terms of $\Delta\mathcal{U}$:

$$V(k) = ||\mathcal{Z}(k) - \mathcal{T}(k)||_{\mathcal{Q}}^2 + ||\Delta\mathcal{U}(k)||_{\mathcal{R}}^2 , \tag{3.6}$$

where:

$$\mathcal{Z}(k) = \begin{bmatrix} \hat{z}(k+H_w|k) \\ \vdots \\ \hat{z}(k+H_p|k) \end{bmatrix} \quad \mathcal{T}(k) = \begin{bmatrix} \hat{r}(k+H_w|k) \\ \vdots \\ \hat{r}(k+H_p|k) \end{bmatrix} \quad \Delta\mathcal{U}(k) = \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix}$$

$$\mathcal{Q} = \begin{bmatrix} Q(H_w) & 0 & \dots & 0 \\ 0 & Q(H_w+1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q(H_p) \end{bmatrix} \quad \mathcal{R} = \begin{bmatrix} R(0) & 0 & \dots & 0 \\ 0 & R(1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R(H_u-1) \end{bmatrix}$$

The controlled output $\mathcal{Z}(k)$ can be expressed as:

$$\mathcal{Z}(k) = \Psi x(k) + \Upsilon u(k-1) + \Theta \Delta \mathcal{U}(k) \ , \tag{3.7}$$

where the matrices $\Psi$, $\Upsilon$ and $\Theta$, which are based on the system matrices $A$, $B$ and $C$, are suitable matrices according to [Maciejowski, 2000, p. 75]. Note, that this work will not dwell in the details of these matrices, as the focus of the thesis is to determine a final form of the control problem for secure cloud computations.

The tracking error is defined as:

$$\mathcal{E}(k) = \mathcal{T}(k) - \Psi x(k) - \Upsilon u(k-1) \ . \tag{3.8}$$

Based on Equation 3.7 and Equation 3.8 the cost function in Equation 3.6 can be rewritten as follows:

$$\begin{aligned} V(k) &= ||\Theta \Delta \mathcal{U}(k) - \mathcal{E}(k)||^2_{\mathcal{Q}} + ||\Delta \mathcal{U}(k)||^2_{\mathcal{R}} && (3.9)\\ &= (\Delta \mathcal{U}(k)^T \Theta^T - \mathcal{E}(k)^T) \cdot \mathcal{Q}(\Theta \Delta \mathcal{U}(k) - \mathcal{E}(k)) + \Delta \mathcal{U}(k)^T \mathcal{R} \Delta \mathcal{U}(k) && (3.10)\\ &= \mathcal{E}(k)^T \mathcal{Q} \mathcal{E}(k) - 2\Delta \mathcal{U}(k)^T \Theta^T \mathcal{Q} \mathcal{E}(k) + \Delta \mathcal{U}(k)^T (\Theta^T \mathcal{Q} \Theta + \mathcal{R}) \Delta \mathcal{U}(k) \ . && (3.11) \end{aligned}$$

Note, that the format of Equation 3.11 corresponds to:

$$\text{const} - \Delta \mathcal{U}(k)^T \mathcal{G}(k) + \Delta \mathcal{U}(k)^T \mathcal{H} \Delta \mathcal{U}(k) \ , \tag{3.12}$$

where:

$$\mathcal{G}(k) = 2\Theta^T \mathcal{Q} \mathcal{E}(k) \ , \qquad\qquad \mathcal{H} = \Theta^T \mathcal{Q} \Theta + \mathcal{R} \ .$$

The optimal control input is found by taking the gradient of the cost function in Equation 3.12, which yields:

$$\begin{aligned} \nabla_{\Delta \mathcal{U}(k)} V(k) &= \frac{\partial V(k)}{\partial \Delta \mathcal{U}(k)} = -\mathcal{G}(k) + 2\mathcal{H} \Delta \mathcal{U}(k) \ , && (3.13)\\ \Delta \mathcal{U}_{opt} &= \frac{1}{2} \mathcal{H}^{-1} \mathcal{G}(k) \ . && (3.14) \end{aligned}$$

Note, that the optimal solution is not guaranteed to be a global optimal. It is therefore necessary to take the second derivative with respect to $\Delta \mathcal{U}(k)$, known as the Hessian. The solution is a global minimum when the Hessian is positive-definite. The Hessian is determined as:

$$H = \frac{\partial^2 V}{\partial \Delta \mathcal{U}(k)^2} = 2\mathcal{H} = 2(\Theta^T \mathcal{Q} \Theta + \mathcal{R}) \ . \tag{3.15}$$

The matrix $\mathcal{H}$ may be ill-conditioned as $\Theta$ often is, thus computing the inverse in Equation 3.14 must be avoided. A suitable method is least-squares which requires the square-root matrices of $\mathcal{Q}$ and $\mathcal{R}$, such that:

$$S_{\mathcal{Q}}^T S_{\mathcal{Q}} = \mathcal{Q} \,, \tag{3.16}$$

$$S_{\mathcal{R}}^T S_{\mathcal{R}} = \mathcal{R} \,, \tag{3.17}$$

where:

| | |
|---|---|
| $S_{\mathcal{Q}}$ | is the square-root matrix of $\mathcal{Q}$, |
| $S_{\mathcal{R}}$ | is the square-root matrix of $\mathcal{R}$. |

The method of determining $S_{\mathcal{Q}}$ and $S_{\mathcal{R}}$ depends on the situation of the matrices. If $\mathcal{Q}$ and/or $\mathcal{R}$ are:

- Diagonal matrices: take the square root of each entry.
- Positive-definite: use the Cholesky algorithm.
- Semi-definite: use Singular Value Decomposition (SVD).

According to [Maciejowski, 2000, p. 77], let

$$M(k) := \begin{bmatrix} S_{\mathcal{Q}}\Theta\Delta\mathcal{U}(k) - \mathcal{E}(k) \\ S_{\mathcal{R}}\Delta\mathcal{U}(k) \end{bmatrix} \,. \tag{3.18}$$

It is claimed, that:

$$V(k) = ||M(k)||^2 \,. \tag{3.19}$$

The correctness of Equation 3.19 is demonstrated by the following derivations:

$$\begin{aligned} \left|\left| \begin{bmatrix} S_{\mathcal{Q}}(\Theta\Delta\mathcal{U}(k) - \mathcal{E}(k)) \\ S_{\mathcal{R}}\Delta\mathcal{U}(k) \end{bmatrix} \right|\right|^2 &= \left|\left| \begin{bmatrix} S_{\mathcal{Q}}(\mathcal{Z}(k) - \mathcal{T}(k)) \\ S_{\mathcal{R}}\Delta\mathcal{U}(k) \end{bmatrix} \right|\right|^2 \\ &= (\mathcal{Z}(k) - \mathcal{T}(k))^T S_{\mathcal{Q}}^T S_{\mathcal{Q}} (\mathcal{Z}(k) - \mathcal{T}(k)) + \Delta\mathcal{U}(k)^T S_{\mathcal{R}}^T S_{\mathcal{R}} \Delta\mathcal{U}(k) \\ &= ||\mathcal{Z}(k) - \mathcal{T}(k)||_{\mathcal{Q}}^2 + ||\Delta\mathcal{U}(k)||_{\mathcal{R}}^2 \\ &= V(k) \,. \end{aligned} \tag{3.20}$$

Taking the square, see Equation 3.20, yields a non-negative cost function, thus its minimum must be zero. The least squares solution that minimizes the cost function $V(k)$ can therefore be written as:

$$\begin{bmatrix} S_{\mathcal{Q}}(\Theta\Delta\mathcal{U}(k) - \mathcal{E}(k)) \\ S_{\mathcal{R}}\Delta\mathcal{U}(k) \end{bmatrix} = 0 \,. \tag{3.21}$$

Equation 3.21 can be rewritten, as:

$$\begin{bmatrix} S_{\mathcal{Q}}\Theta \\ S_{\mathcal{R}} \end{bmatrix} \Delta\mathcal{U}(k) = \begin{bmatrix} S_{\mathcal{Q}}\mathcal{E}(k) \\ 0 \end{bmatrix} . \tag{3.22}$$

Note, that Equation 3.22 has the form $Ax = b$ and can be solved as a linear system. The solution is the predicted optimal control input $\Delta\mathcal{U}(k)$. It is desired to compute the solution securely, hence standard operations cannot be utilized. Considerations on how to obtain a secure solution is presented in chapter 4.

## 3.3 Constrained MPC$_{\text{ontrol}}$

MPC$_{\text{ontrol}}$ with constraints is a more realistic control problem as the constraints serves the purpose of attaining a satisfying dynamic performance of a system.
It is desirable to operate close to the constraints to optimize performance, however, with a margin to allow unexpected disturbances to be handled. Constraints on actuator slew rate, actuator range and on the controlled variable are expressed in Equation 3.23, 3.24 and 3.25, respectively.

$$E \begin{bmatrix} \Delta\mathcal{U}(k) \\ 1 \end{bmatrix} \leq 0 \qquad \text{where} \quad \Delta\mathcal{U}(k) = \left[\Delta\hat{u}(k|k)^T ... \Delta\hat{u}(k+H_u-1|k)^T\right]^T , \tag{3.23}$$

$$F \begin{bmatrix} \mathcal{U}(k) \\ 1 \end{bmatrix} \leq 0 \qquad \text{where} \quad \mathcal{U}(k) = \left[\hat{u}(k|k)^T ... \hat{u}(k+H_u-1|k)^T\right]^T , \tag{3.24}$$

$$G \begin{bmatrix} \mathcal{Z}(k) \\ 1 \end{bmatrix} \leq 0 \qquad \text{where} \quad \mathcal{Z}(k) = \left[\hat{z}(k+H_w|k)^T ... \hat{z}(k+H_p|k)^T\right]^T . \tag{3.25}$$

Similar to the cost function that was reformulated in the previous section the constraints must also be reformulated in terms of $\Delta\mathcal{U}$. The reader is referred to [Maciejowski, 2000, p. 81-83] for the derivations. The resulting constraints are the following:

$$\begin{bmatrix} \mathcal{F} \\ \Gamma\Theta \\ W \end{bmatrix} \Delta\mathcal{U}(k) \leq \begin{bmatrix} -\mathcal{F}_1 u(k-1) - f \\ -\Gamma(\Psi x(k) + \Upsilon u(k-1)) - g \\ w \end{bmatrix} . \tag{3.26}$$

The constraints in Equation 3.26 are linear inequality constraints to the quadratic minimization problem:

$$\min_{\Delta\mathcal{U}(k)} V(k) = -\Delta\mathcal{U}(k)^T\mathcal{G} + \Delta\mathcal{U}(k)^T\mathcal{H}\Delta\mathcal{U}(k) , \tag{3.27}$$

which constitutes a convex optimization problem.

Solving constrained $\text{MPC}_{\text{ontrol}}$ problems is most often done using software solvers such as Yalmip or the CVX toolbox in MATLAB. It is not feasible to utilize these solvers in a secure way, hence solving the optimization problem must be done differently.

The constrained $\text{MPC}_{\text{ontrol}}$ optimization problem can be formulated as a classic Quadratic Programming (QP) problem according to [Maciejowski, 2000, p. 84-86].

A standard QP problem is given as:

$$\min_{\theta} \quad f(\theta) = \frac{1}{2}\theta^T \Psi \ \theta + \psi^T \ \theta \ , \tag{3.28}$$

s.t:

$$g(\theta) = \Omega_a \theta - \omega_a \ . \tag{3.29}$$

Assuming that the constraints are active constraints and hence equality constraints, the optimization problem can be solved using Lagrange multipliers.

The optimization problem then becomes:

$$\min_{\theta,\lambda} \quad L(\theta, \lambda) \ , \tag{3.30}$$

where:

$$\begin{aligned} L(\theta, \lambda) &= f(\theta) + \lambda(g(\theta)) \\ &= \frac{1}{2}\theta^T \Psi \theta + \psi^T \theta + \lambda(\Omega_a \theta - \omega_a) \ . \end{aligned}$$

The partial derivatives are:

$$\nabla_\theta L(\theta, \lambda) = \Psi\theta + \psi + \Omega_a^T \lambda \ , \tag{3.31}$$

$$\nabla_\lambda L(\theta, \lambda) = \Omega_a \theta - \omega_a \ . \tag{3.32}$$

Equation 3.31 and 3.32 can be put on matrix form:

$$\begin{bmatrix} \Psi & \Omega_a^T \\ \Omega_a & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \lambda \end{bmatrix} = \begin{bmatrix} -\psi \\ \omega_a \end{bmatrix} \ . \tag{3.33}$$

It is seen that Equation 3.33 is a linear system similarly to the unconstrained case thus solving either case securely can be obtained by the same approach. Note, that the set of active constraints will change in practice thus switching between multiple linear controllers must be utilized [Maciejowski, 2000, p. 86-88].

The focus of the thesis is delimited to investigate how control inputs can be computed securely thus further details on active constrained $\text{MPC}_{\text{ontrol}}$ will not be given.

This chapter has presented how unconstrained and equality constrained $\text{MPC}_{\text{ontrol}}$ problems can be expressed as linear systems. Considerations on how to solve linear systems securely are presented in the following chapter.

# 4 | Secure Solution

Chapter 3 concludes that both unconstrained and equality constrained $\text{MPC}_{\text{ontrol}}$ problems can be formulated as linear systems. Determining a solution to such systems must be done securely using $\text{MPC}_{\text{omp}}$. It is important to consider what information within the control algorithms that must be kept private. Different levels of privacy can be chosen depending on the content of the data being handled. The levels of privacy can be:

**High**: private plant dynamics, private states, private control inputs.

**Medium**: private states, private control inputs.

**Low**: private control inputs.

The level of privacy chosen to study is the maximum level where no information of the system nor its control actions are disclosed.

This is not a trivial task, as a number of parties each hold data that must be kept private, while at the same time collaborating to compute shares of the solution. Note, that it is only the autonomous vehicle being the data owner, that must learn the result of the secure computations. How this can be achieved is presented in this chapter.

This chapter describes the linear system of the unconstrained case. First $[\![\mathbf{A}]\!]$, hereafter referred to as the input matrix, and $[\![\mathbf{b}]\!]$, hereafter referred to as the observation vector, are determined securely. Afterwards a generic approach for solving a linear system is presented. The method is applicable for both the unconstrained and the equality constrained case.

## 4.1 Secure Unconstrained Case

Recap, that the optimal control input can be formulated on the standard form $\mathbf{A}\,\mathbf{x} = \mathbf{b}$, see Equation 3.22 in chapter 3, where:

$\mathbf{A}$ corresponds to $\mathbf{S}_{\mathcal{Q}}\mathbf{\Theta}$,

$\mathbf{x}$ corresponds to $\Delta\mathcal{U}(k)$,

$\mathbf{b}$ corresponds to $\mathbf{S}_{\mathcal{R}}\mathcal{E}(k)$.

The solution to the linear system is the control input $\Delta\mathcal{U}(k)$ that must be kept private according to the chosen level of privacy. The collaborating parties that compute $\Delta\mathcal{U}(k)$ are therefore not allowed to learn the result. To ensure this, the

parties must return their individual shares of the result to the data owner. The data owner then reconstructs the private result and is the only one learning it.

The square-root matrices $\mathbf{S}_{\mathcal{Q}}$ and $\mathbf{S}_{\mathcal{R}}$ are formed based on the weighting matrices $\mathcal{Q}$ and $\mathcal{R}$, respectively. If the square-root matrices are not diagonal matrices either the Cholesky algorithm or an SVD approach must be utilized. This is tricky to compute securely. It is deemed acceptable to compute the square-root matrices openly as the penalties on the states and control inputs do not reveal information of great value in itself. Note, that this may not always be the case.

By having derived the square-root matrices openly, it is possible to securely determine the input matrix $\mathbf{A}$ and the observation vector $\mathbf{b}$.

The input matrix $\mathbf{A}$ is the product $\mathbf{S}_{\mathcal{Q}}\mathbf{\Theta}$, where $\mathbf{\Theta}$ is part of the system dynamics and must be kept private. For a matrix to be private all its entries must be private:

$$[\![\mathbf{A}]\!] = \begin{bmatrix} [\![a_{11}]\!] & [\![a_{12}]\!] \\ [\![a_{21}]\!] & [\![a_{22}]\!] \end{bmatrix} . \tag{4.1}$$

The $[\![\mathbf{A}]\!]$ matrix is obtained by securely computing $\mathbf{S}_{\mathcal{Q}}[\![\mathbf{\Theta}]\!]$ using *Beaver's triplet* multiplication, see Protocol 2.4 in chapter 2.

Note, that the secure multiplication method can handle matrices if the *triplets* are modified. The *triplets* must be matrices of similar dimensions as the matrices they are multiplied by. The secure *Beaver's triplet* multiplication method is modified as follows:

$$[\![\mathbf{AB}]\!] = \mathbf{DE} + \mathbf{D}[\![\beta]\!] + [\![\alpha]\!]\mathbf{E} + [\![\gamma]\!] , \tag{4.2}$$

where $[\![\alpha]\!]$, $[\![\beta]\!]$ and $[\![\gamma]\!]$ form a *triplet* of matrices. $[\![\mathbf{A}]\!]$, $[\![\mathbf{B}]\!]$, $\mathbf{D}$ and $\mathbf{E}$ are matrices corresponding to the $[\![a]\!]$, $[\![b]\!]$, $d$ and $e$ variables in the *Beaver's triplet* multiplication protocol. Equation 2.34-2.36 in chapter 2 demonstrate the correctness of the *Beaver's triplet* multiplication method. The derivations are true for matrix operations thus matrix multiplication using *Beaver's triplets* is valid. Note, that as matrices do not commute the multiplication arrangement in Equation 4.2 is specific. A two dimensional example is presented in Appendix B to demonstrate the method.

The $\mathbf{b}$ vector is the product $\mathbf{S}_{\mathcal{R}}\mathcal{E}(k)$, where the tracking error $\mathcal{E}(k) = \mathcal{T}(k) - \Psi x(k) - \Upsilon u(k-1)$ must be computed securely as it depends on the private states, the private the previous control inputs and the private system dynamics. The protocol for secure computation of the observation vector $[\![\mathbf{b}]\!]$ is listed in Protocol 4.1.

**Protocol 4.1** Solving $[\![\mathbf{b}]\!] = \mathbf{S}_{\mathcal{R}} \cdot [\![\mathcal{E}(k)]\!]$

**Input:** The states $[\![\mathbf{x}(k)]\!]$, the previous control inputs $[\![\mathbf{u}(k-1)]\!]$, the system references $[\![\mathcal{T}(k)]\!]$, the system dynamics $[\![\Upsilon]\!]$, $[\![\Psi]\!]$ and the open square-matrix $\mathbf{S}_{\mathcal{R}}$.

**Output:** The shares $[\![b_1, ..., b_n]\!]$ of $[\![\mathbf{b}]\!]$.

1: $[\![\Psi \mathbf{x}_{prod}]\!] = [\![\Psi]\!] \cdot [\![\mathbf{x}(k)]\!]$
2: $[\![\Upsilon u_{prod.}]\!] = [\![\Upsilon]\!] \cdot [\![\mathbf{u}(k-1)]\!]$
3: $[\![\mathcal{E}(k)]\!] = [\![\mathcal{T}(k)]\!] - [\![\Psi x_{prod}]\!] - [\![\Upsilon u_{prod}]\!]$
4: $[\![\mathbf{b}]\!] = \mathbf{S}_{\mathcal{R}} \cdot [\![\mathcal{E}(k)]\!]$

In chapter 2 only secure operations for addition and multiplication were presented as these can describe any polynomial. The subtraction in Protocol 4.1 line 3 remains secure due to the following.
Let $s_a$ and $s_b$ be two secrets distributed among $n$ parties, then:

$$[\![s_a]\!] = [\![s_{a1}, s_{a2}, \ldots, s_{an}]\!] , \tag{4.3}$$

$$[\![s_b]\!] = [\![s_{b1}, s_{b2}, \ldots, s_{bn}]\!] , \tag{4.4}$$

$$[\![s_a]\!] - [\![s_b]\!] = [\![s_{a1} - s_{b1}, s_{a2} - s_{b2}, \ldots, s_{an} - s_{bn}]\!] . \tag{4.5}$$

The result of reconstructing the local subtractions of shares is equivalent to the subtraction of the two secrets.

As the derivations of $[\![\mathbf{A}]\!]$ and $[\![\mathbf{b}]\!]$ have been determined, a solution to the linear system must be computed. This is non-trivial to do as it must be done securely.

A paper [Bouman and Vreede, 2018] on secure linear algebra suggests a method for solving $[\![\mathbf{A}]\!][\![\mathbf{x}]\!] = [\![\mathbf{b}]\!]$ over a finite field in $\text{MPC}_{\text{omp}}$ using Guassian elimination. It is deemed suitable to solve the $\text{MPC}_{\text{ontrol}}$ problems of the thesis and is introduced in the following section.

## 4.2 Solving Linear Systems Securely

The objective is to solve $[\![\mathbf{A}]\!] \cdot [\![\mathbf{x}]\!] = [\![\mathbf{b}]\!]$ for $[\![\mathbf{x}]\!]$, where $[\![\mathbf{A}]\!] \in \mathbb{F}_p^{m \times n}$ and $[\![\mathbf{b}]\!] \in \mathbb{F}_p^{n \times \ell}$. Gaussian elimination can be used to solve a linear system by shifting and reducing the rows until it becomes row echelon form.

Row echelon form is defined as the resulting matrix from Gaussian elimination, where zero-rows are placed last in the matrix.

Moreover the pivot of each row must be on the left hand side of the leading entry, also known as the pivot of the row below it [Weisstein, 2018], as:

$$\begin{bmatrix} a_1 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 \\ 0 & 0 & 0 & a_3 \end{bmatrix} .$$

Note, that the leading entries can be any values as it is not reduced row echelon form that otherwise requires the entries to be 1.

The protocol suggested in [Bouman and Vreede, 2018] can securely solve linear systems without pivoting for both over- and underdetermined systems of unknown rank. This is an important feature as it decreases the computational complexity that otherwise occurs when solved using $\text{MPC}_{\text{omp}}$. The secure protocol is moreover capable of solving for multiple $[\![\mathbf{b}]\!]$ vectors [Bouman and Vreede, 2018]. This is however not pursued as it is simply an extension of the same principles.

A precondition to consider prior to utilizing the protocol suggested in [Bouman and Vreede, 2018] is that the linear system must be solvable, which requires:

$$\text{rank } [\![\mathbf{A}]\!] = \text{rank } [\![\mathbf{A}]\!] \mid [\![\mathbf{b}]\!] . \tag{4.6}$$

Moreover to solve a linear system by Gaussian elimination without pivoting, the $[\![\mathbf{A}]\!]$ matrix must have generic rank profile [Bouman and Vreede, 2018]. To increase the probability of satisfying this precondition, Toeplitz matrices can advantageously be used to modify the system such that $[\![\mathbf{A}']\!] = \mathbf{U}[\![\mathbf{A}]\!]\mathbf{L}$ and $[\![\mathbf{b}']\!] = \mathbf{U}[\![\mathbf{b}]\!]$. A lemma states that the probability that $[\![\mathbf{A}']\!] = \mathbf{U}[\![\mathbf{A}]\!]\mathbf{L}$ has generic rank profile is greater than $1-\mu(\mu+1)/p$ where $\mu = \min(m, n)$ and $p$ is the cardinality of $\mathbb{F}_p$ [Bouman and Vreede, 2018]. Note, that as the protocol does not require the rank to be known, $\mu$ is an upper bound for the rank of $[\![\mathbf{A}]\!]$.

The Toeplitz matrices take form as follows:

$$\mathbf{U} := \begin{bmatrix} 1 & u_2 & u_3 & \dots & u_m \\ 0 & 1 & u_2 & \dots & u_{m-1} \\ 0 & 0 & 1 & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & u_2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} , \qquad \mathbf{L} := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \ell_1 & 1 & 0 & 0 & 0 \\ \ell_3 & \ell_2 & 1 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \ell_n & \ell_{n-1} & \dots & \ell_2 & 1 \end{bmatrix} . \tag{4.7}$$

The Toeplitz matrices $\mathbf{U}$ and $\mathbf{L}$ are constructed such that the non-zero entries $u_n$ and $\ell_n$, hereafter referred to as the Toeplitz entries, are random independent uniformly distributed variables within the finite field $\mathbb{F}_p$ [Kaltofen and Saunders, 1991].

Generic rank profile is ensured if the leading principal minor of $[\![\mathbf{A}']\!]$ for all $k \in [r]$ are non-zero [Bouman and Vreede, 2018], where $r$ is the rank of $[\![\mathbf{A}']\!]$. The leading principal minor is the determinant of a minor where $n-k$ rows and columns have been removed, as:

$$\det [\![\mathbf{A}'_{n-k}]\!] \qquad \text{where} \qquad [\![\mathbf{A}'_{n-k}]\!] = \begin{bmatrix} [\![a'_{1,1}]\!] & \dots & [\![a'_{1,n-k}]\!] \\ \vdots & \ddots & \vdots \\ [\![a'_{n-k,1}]\!] & \dots & [\![a'_{n-k,n-k}]\!] \end{bmatrix}.$$

To exemplify this the leading principal minors of a $3 \times 3$ matrix with rank $r = 2$ thus $k = 1, 2$ are presented.

For $k = 1$:

$$\det [\![\mathbf{A}'_2]\!] \qquad \text{where} \qquad [\![\mathbf{A}'_2]\!] = \begin{bmatrix} [\![a'_{11}]\!] & [\![a'_{12}]\!] \\ [\![a'_{21}]\!] & [\![a'_{22}]\!] \end{bmatrix}.$$

For $k = 2$:

$$\det [\![\mathbf{A}'_1]\!] \qquad \text{where} \qquad [\![\mathbf{A}'_1]\!] = [\![a'_{11}]\!].$$

If non of the leading principal minors are zero the precondition is satisfied.

Before utilizing the protocol suggested in [Bouman and Vreede, 2018] securely, it is computed openly in MATLAB to verify its correctness. Open computations are defined by not using $\text{MPC}_{\text{omp}}$ hence no privacy is preserved.

The protocol has proven to yield the correct output when computed openly and is thus verified. It will be utilized to securely solve private linear systems within the thesis. The secure protocol is seen in Protocol 4.2 on the following page and is hereafter referred to as the secure GE protocol.

**Protocol 4.2** Secure GE protocol [Bouman and Vreede, 2018]

**Input:** $[\![\mathbf{A}]\!] \in \mathbb{F}_p^{m\times n}$ , $[\![\mathbf{b}]\!] \in \mathbb{F}_p^{n\times \ell}$, Toeplitz matrices $\mathbf{U}$, $\mathbf{L} \in \mathbb{F}_p^{m\times n}$ and $[\![h]\!] = [\![1]\!], [\![t]\!] = [\![1]\!]$.
**Output:** The shares $[\![\mathbf{x}_1, ..., \mathbf{x}_n]\!]$ of $[\![\mathbf{x}]\!] \in \mathbb{F}_p^{n\times \ell}$.

1: $$[\![\mathbf{C}]\!] = \begin{bmatrix} \mathbf{I}_{\mu\times m}\mathbf{U}[\![\mathbf{A}]\!]\mathbf{L} & \mathbf{I}_{\mu\times m}\mathbf{U}[\![\mathbf{b}]\!] \\ [\mathbf{I}_n] & \mathbf{0}^{n\times\ell} \end{bmatrix} \qquad \text{where } \mu = \min(m, n)$$

2: **for** $k$ in range $(0, \mu)$:
3: $\quad [\![r_k]\!] = ([\![c_{k,k}]\!] \overset{?}{\neq} 0)$
4: $\quad [\![c_{\mu+k,k}]\!] = [\![h]\!]$
5: $\quad [\![f_k]\!] = [\![h]\!]$
6: $\quad [\![t]\!] = [\![t]\!] \cdot [\![h]\!]$
7: $\quad [\![h]\!] = [\![h]\!] \cdot ([\![c_{k,k}]\!] + 1 - [\![r_k]\!])$
8: $\quad$ **for** $i$ in range $(0, \mu + k)$:
9: $\quad\quad$ **for** $j$ in range $(k + 1, n + \ell)$:
10: $\quad\quad\quad$ **if** $i \neq k \wedge (i \leq \mu \vee j \leq n)$:
11: $$[\![c_{i,j}]\!] = \Big(([\![c_{k,k}]\!] + 1 - [\![r_k]\!]) \quad [\![c_{k,j}]\!]\Big) \cdot \begin{pmatrix} [\![c_{i,j}]\!] \\ -[\![c_{i,k}]\!] \end{pmatrix}$$

12: $$\begin{bmatrix} \\ [\![\mathbf{x}]\!] \\ \\ \end{bmatrix} = [\![\mathbf{C}]\!] \qquad \text{where } [\![\mathbf{x}]\!] \in \mathbb{F}_p^{n\times\ell}$$

13: $[\![g]\!] = ([\![t]\!] \cdot [\![h]\!])^{-1}$
14: $[\![\mathbf{x}]\!] = [\![g]\!] \cdot [\![t]\!]\ \mathbf{L}\ \mathbf{I}_{n\times\mu}\text{diag}([\![\mathbf{f}]\!])\ [\![\mathbf{x}]\!]$
15: **return** $[\![\mathbf{x}]\!]$

Note, that the protocol is presented with zero based indexing and inclusive-exclusive range syntax.
The inputs to the protocol are the private $[\![\mathbf{A}]\!]$ and $[\![\mathbf{b}]\!]$, two variables $[\![h]\!]$ and $[\![t]\!]$ and lastly the open Toeplitz matrices $\mathbf{U}$ and $\mathbf{L}$.

The secure GE protocol must be simulated with multiple threads to test its utility before implementing it on a real network of multiple servers. The simulation implementation of the secure GE protocol is described in the following chapter.

# 5 | Simulation

This chapter describes in detail the multi thread simulation implementation of the most significant computations of the secure GE protocol, see Protocol 4.2 in chapter 4.

To verify the simulation implementation each secure computation is compared to the corresponding open MATLAB computation, which has previously proved to yield correct results, see chapter 4. To be able to compare the result of each secure computation with its corresponding open computation in MATLAB, it is necessary to broadcast and reconstruct the result of each computation from the secure GE protocol. This is a preventive measure as the simulation invokes multiple sub protocols in multiple threads simultaneously that must sync in order to achieve correct shares.

The multi thread simulation of the secure GE protocol is implemented as Python3 programming. The virtual setup is one data owner and three cloud servers. The simulation implementation is limited to handle square input matrices of order 2.
Note, that the simulation implementation has been successfully implemented and the code can be accessed on: https://github.com/avtl/thesis/simulation. The most interesting computations within the secure GE protocol in regards to $\text{MPC}_{\text{omp}}$ is discussed in detail as follows.

## 5.1 Secure Equality Test

**Secure GE protocol, line 3:** $[\![r_k]\!] = ([\![c_{k,k}]\!] \stackrel{?}{\neq} 0)$

The protocol invokes equality testing. The paper [Bouman and Vreede, 2018] that suggests the secure GE protocol refers to the paper [Nishide and Ohta, 2007] for secure equality testing. This is, however, a rather complex method. Considering the context that the equality testing must be used within, it can be justified to implement a simpler method. It is essential not to reveal the true value of the entry $[\![c_{k,k}]\!]$ as it is based on the private input matrix $[\![A]\!]$ and the open Toeplitz matrices, see line 1 in Protocol 4.2. The result of the equality test can be argued to have very little value, thus this information can be disclosed without compromising the privacy of the private inputs. Note, that the information revealed is the equality test results of the $1, \ldots, \mu$ diagonal entries of $[\![\mathbf{C}]\!]$.

A method for semi-secure equality testing is implemented by having the data owner distribute shares of a secret positive random variable $[\![d]\!]$. If the entry $[\![c_{k,k}]\!]$ is different from zero, multiplying it by a secret positive random variable will not affect the result of the equality test. The product of $[\![c_{k,k}]\!] \cdot [\![d]\!]$ is then broadcast and reconstructed. If the equality is true it returns 1 and 0 otherwise.

The result of the equality test is split into shares and distributed to the cloud servers. This is superfluous as the result is reconstructed based on broadcast values. It is, however, implemented this way as it mimics an ideal implementation, though it is for now disclosed information.

## 5.2 Subtracting Shares from an Open Value

**Secure GE protocol, line 7:** $[\![h]\!] = [\![h]\!] \cdot ([\![c_{k,k}]\!] + 1 - [\![r_k]\!])$

First consider the operations within the parentheses. The boolean array $r_k$ stores the result from the equality test in line 3. Note, that it is not private due to a semi-secure equality test. If the entry $[\![c_{k,k}]\!]$ is different from zero, $r_k$ is one. When subtracted from one, no contribution is added to the entry $[\![c_{k,k}]\!]$. However, if $[\![c_{k,k}]\!]$ is zero, the equality test in line 3 returns false and $r_k$ will instead be zero. This ensures that whenever the entry $[\![c_{k,k}]\!]$ is zero, the $[\![h]\!]$ variable will not be multiplied by zero. Note, that $r_k$ can be subtracted from 1 though 1 is not split into shares, based on the assertion:

$$1 \stackrel{?}{=} [\![1]\!] \quad \Rightarrow \quad \text{True} \, . \tag{5.1}$$

The coefficients of the shares' polynomial are chosen randomly within a finite field $\mathbb{F}_p$. Recall that a share polynomial is defined as:

$$f(x) = a + a_1 x + \ldots + a_t x^t \, , \tag{5.2}$$

where the secret $s$ is the polynomial evaluated at zero, $s = f(0) = a$. In principle, all coefficients can be zero except $a$ that has to be the secret, as:

$$f(x) = 1 + 0 \cdot x + \ldots + 0 \cdot x^t \, . \tag{5.3}$$

Evaluating Equation 5.3 to different values does not change the function output that will always be 1.
An example of three shares illustrates this:

$$f(1) = 1 \, , \tag{5.4}$$
$$f(2) = 1 \, , \tag{5.5}$$
$$f(3) = 1 \, . \tag{5.6}$$

Reconstructing the secret of Equation 5.3 based on Equation 5.4-5.6 is redundant, as it is easily seen that the secret is 1 as $s = f(0) = a = 1$. It is however reconstructed by Lagrange interpolation using Equation 2.7 from chapter 2, to demonstrate the correctness of the assertion.

$$\begin{aligned} &\frac{-2}{1-2} \cdot \frac{-3}{1-3} \cdot 1 + \frac{-1}{2-1} \cdot \frac{-3}{2-3} \cdot 1 + \frac{-1}{3-1} \cdot \frac{-2}{3-2} \cdot 1 \\ &= 3 - 3 + 1 = 1 \, . \end{aligned} \tag{5.7}$$

From Equation 5.7 it can be seen that treating $[\![1]\!]$ simply as shares of all 1 is in fact accurate.

Ultimately the two shares within the parentheses can be summed locally by each party without distorting the result of the later reconstruction, as presented:

$$[\![s_a]\!] = [\![s_{a1}, s_{a2}, \ldots, s_{an}]\!] , \tag{5.8}$$

$$[\![s_b]\!] = [\![s_{b1}, s_{b2}, \ldots, s_{bn}]\!] , \tag{5.9}$$

$$[\![s_a]\!] + [\![s_b]\!] = [\![s_{a1} + s_{b1}, s_{a2} + s_{b2}, \ldots, s_{an} + s_{bn}]\!] , \tag{5.10}$$

where $s_a$ and $s_b$ are two secrets distributed among $n$ parties.
When the content within the parentheses is derived it must be multiplied with $[\![h]\!]$. Secure multiplication is computed using the *Beaver's triplet* multiplication protocol, see Protocol 2.4 in chapter 2.

## 5.3 Secure Inversion

**Secure GE protocol, line 13:** $[\![g]\!] = ([\![t]\!] \cdot [\![h]\!])^{-1}$

Another computation that is not trivial to implement is the inversion of the product of two shares that are multiplied using the *Beaver's triplet* multiplication protocol. Note, that the inverse of each share does not comply with the inverse of the secret when reconstructed:

$$[\![s]\!]^{-1} \neq [\![s_1^{-1}, s_2^{-1}, \ldots, s_n^{-1}]\!] . \tag{5.11}$$

Within the *SECURE* research project a method for secure inversion has been investigated and implemented by [Tjell, 2018]. It inverts a private variable by integer division based on bit sequences. Integer division means that two integers are divided and the result is rounded to the nearest integer. Before transforming the result into an integer it must be scaled in order to limit the precision loss. As the shares are private, it is impossible to ensure appropriate scaling. This is defined as blind scaling. As the method is vulnerable, already known to the *SECURE* project and based on bit sequences that have not been investigated in the thesis, this method is not utilized for secure inversion.

The paper [Bouman and Vreede, 2018] that suggests the secure GE protocol refers to the paper [Bar-Ilan and Beaver, 1989] for a secure inversion method. The inversion method presented in [Bar-Ilan and Beaver, 1989] securely multiplies the secret, which must be inverted, with a private random variable different from zero and broadcasts the result. It is afterwards openly inverted in the field of rational numbers $\mathbb{Q}$. The secret is not revealed as the random variable is private thus its effect on the value that is broadcast is unknown. The inversion method is seen in Protocol 5.1. The inputs are the secret $[\![s]\!]$, in this case equivalent to the private product of $[\![t]\!] \cdot [\![h]\!]$, and the private random variable $[\![z]\!]$.

**Protocol 5.1** Secure inversion protocol

**Input:** A secret $[\![s; f_s]\!]_t$, a random variable $[\![z; f_z]\!]_t$ and a scaling factor $c_s$.

**Output:** The inverted scaled secret $[\![s^{-1}; f_s]\!]_t \cdot c_s = [\![s_c^{-1}; f_{sc}]\!]_t$.

1: Party $P_i$ computes

$$[\![w_i; f_{wi}]\!]_t = [\![s_i; f_{si}]\!]_t \cdot [\![z_i; f_{zi}]\!]_t \ , \tag{5.12}$$

using the *Beaver's triplet* multiplication protocol.

2: Party $P_i$ broadcasts $[\![w_i; f_{wi}]\!]_t$ .

3: Mapping

$$\begin{aligned} f : \mathbb{F}_p \rightarrow \mathbb{Q} \ , \\ f([\![w_i; f_{wi}]\!]_t) = w \ , \end{aligned} \tag{5.13}$$

by reconstructing $w$ from the broadcast shares.

4: Compute $w^{-1}$ openly.

5: Scale and round to nearest integer as: $k = [w^{-1} \cdot c_s]$.

6: Mapping

$$\begin{aligned} g : \mathbb{Q} \rightarrow \mathbb{F}_p \ , \\ g(k) = (k_i; f_{ki})_t \ , \end{aligned} \tag{5.14}$$

where $(k_i; f_{ki})_t$ is the $i^{\text{th}}$ share of $k$.
Note, that as $k$ is not secret parentheses are used instead of brackets.

7: Each party holds $(k_i; f_{ki})_t$.

8: Party $P_i$ computes

$$[\![s_{si}^{-1}; f_{si}]\!]_t = (k_i; f_{ki})_t * [\![z_i; f_{zi}]\!]_t \ , \tag{5.15}$$

using the *Beaver's triplet* multiplication protocol.

The following derivations show that multiplying the random variable $[\![z]\!]$ with the inverted, scaled and rounded variable $k$ yields the inverted scaled secret $[\![s_s^{-1}]\!]$.

Note that the notation is simplified, such that Equation 5.15 is equivalent to Equation 5.16.

$$\begin{aligned} [\![s_s^{-1}]\!] &= k * [\![z]\!] && (5.16)\\ &= [c_s \cdot w^{-1}] * [\![z]\!] && (5.17)\\ &= c_s \cdot ([\![s]\!] * [\![z]\!])^{-1} * [\![z]\!] && (5.18)\\ &= c_s \cdot [\![s^{-1}]\!] * [\![z^{-1}]\!] * [\![z]\!] && (5.19)\\ &= c_s \cdot [\![s^{-1}]\!] = [\![s_s^{-1}]\!] && (5.20) \end{aligned}$$

The inversion variable $w$ maps from the finite field $\mathbb{F}_p$ to the field of rational numbers $\mathbb{Q}$ and to the finite field $\mathbb{F}_p$ again, presented as follows:

$$\begin{aligned} \mathbb{F}_p &\longrightarrow \mathbb{Q} \longrightarrow \mathbb{Q} \longrightarrow \mathbb{F}_p \\ [\![w]\!] &\longrightarrow w \longrightarrow [w_s^{-1}] \longrightarrow k \end{aligned} \tag{5.21}$$

From Equation 5.21 it is seen that the private variable $[\![w]\!]$ is mapped from $\mathbb{F}_p$ by broadcasting and reconstruction, such that it becomes an open value within $\mathbb{Q}$. It is then inverted by division and scaled to minimize precision loss before it is rounded to nearest integer. The inverted, scaled and rounded variable $k = [w_s^{-1}]$ is then mapped from $\mathbb{Q} \to \mathbb{F}_p$ where it is used within the secure GE protocol.

To ensure that the mappings are injective, the following conditions must be satisfied:

$$w \leq p - 1 \ : \mathbb{F}_p \to \mathbb{Q} \ , \tag{5.22}$$

$$[w_s^{-1}] \leq p - 1 \ : \mathbb{Q} \to \mathbb{F}_p \ , \tag{5.23}$$

where $p$ is the cardinality prime of the finite field $\mathbb{F}_p$. If the conditions are not satisfied wrap-around occurs. As $\text{MPC}_{\text{omp}}$ works within the finite field $\mathbb{F}_p$, wrap-around is defined as when a value is larger than $p - 1$.
An example, where wrap-around causes a mapping to be not injective, is presented as follows:

$$f : \mathbb{Q} \to \mathbb{F}_5 \ , \tag{5.24}$$

$$f(6) = 1 \ ,$$

$$g : \mathbb{F}_5 \to \mathbb{Q} \ , \tag{5.25}$$

$$f(1) = 1 \ .$$

The mapping in Equation 5.24 is not injective, as the mapping in Equation 5.25 does not equal 6.

The product $w$ of the random variable $[\![z]\!]$ and the secret $[\![s]\!]$ must satisfy the condition for injective mapping $\mathbb{F}_p \to \mathbb{Q}$, see Equation 5.22, as it shall be inverted

by division in the field of rational numbers $\mathbb{Q}$. As the data owner that distributes shares of its private data knows $[\![s]\!]$, it can determine an upper bound for the range, which the random uniformly distributed $[\![z]\!]$ variable is generated within, such that $w \leq p - 1$. This will ensure, that the open value $w$ does not wrap-around and that the mapping $\mathbb{F}_p \rightarrow \mathbb{Q}$ will be injective.

To satisfy the condition for injective mapping $\mathbb{Q} \rightarrow \mathbb{F}_p$, see Equation 5.23, the scaling factor $c_s$ must be smaller than $p - 1$ as any inversion will yield a value less than 1, hence the scaled outcome cannot exceed $p - 1$. The precision of the scaled inverted variable $[w_s^{-1}]$ is compromised and therefore forwards precision loss to the inverted scaled secret $[\![s_s^{-1}]\!]$. It is, however, preferable in contrast to a faulty inversion result due to wrap-around when mapping to the finite field $\mathbb{F}_p$.

Protocol 5.1 is an insufficient inversion method as it operates within multiple fields, where a small scaling factor is required to ensure injective mapping. The fixed scaling introduces precision loss and Protocol 5.1 thus fails to provide exact inverting.

As the inversion method utilized in [Tjell, 2018] was found to be vulnerable due to blind scaling and Protocol 5.1 suffers precision loss, it is preferable to utilize another inversion method. It is, however, left for future work to research a more precise inversion method within finite field arithmetic as the thesis investigates the secure GE protocol in a control context. It is therefore chosen to invoke Protocol 5.1 for secure inversion despite the precision loss. The precision loss ultimately affects the output of the secure GE protocol. This will be considered and discussed in chapter 7.

This completes the description of the simulation implementation of the secure GE protocol that solves linear systems securely.

The secure GE protocol and its subprotocols must be implemented on multiple servers that constitute a real network. This is presented in the following chapter.

# 6 | Implementation

The secure GE protocol for solving linear systems securely must be implemented on a network of multiple parties to prove its feasibility. Note, that the communication channels between the parties are assumed to be secure based on appropriate encryption. Data encryption is a task of great complexity in itself, thus it is not realized in the implementation as the focus of the thesis is secure $\text{MPC}_{\text{omp}}$ in a control context. The network setup is firstly described followed by a visual presentation of the implemented network.

## 6.1 Network Setup

The implemented $\text{MPC}_{\text{omp}}$ network consists of one data owner being the autonomous vehicle and three cloud servers that must be fully connected. The data owner holds an unsolved private linear system and must obtain the solution without any other party learning it. The process is illustrated in Figure 6.1.

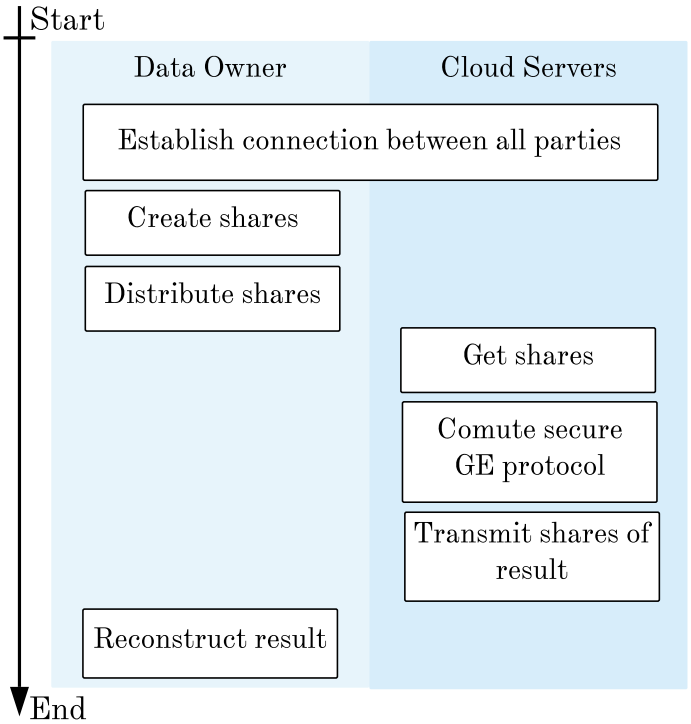


**Figure 6.1:** Conceptual diagram of the actions in relation to each other between the data owner and the parrallel cloud servers.

First connection between all parties must be established. Then the data owner distributes shares of its linear system to the cloud servers. It moreover distributes shares of a random inversion variable and *Beaver's triplets* for secure multiplications.

Note, that as this network only has one data owner to compute secure solutions for, the data owner can generate and distribute *Beaver's triplets* itself. In a real life implementation with many autonomous vehicles, an individual server must generate and distribute the *Beaver's triplets* to the cloud servers, as the *triplets* are private.

When the cloud servers have received shares from the data owner they compute the secure GE protocol and returns their share of the result to the data owner. The data owner then reconstruct the secret that is the solution to the linear system. Figure 6.1 illustrates how the different actions are arranged according to when they occur between the data owner and the parallel cloud servers.

Each party in the network must be a computing device. The Raspberry Pi (RP) single board computers are chosen to constitute the computing servers of the network. The RPs each have a display interface to visualize the secure computational process of solving linear systems between the parties.
Different from the multi thread simulations in chapter 5 the RP network must actual communicate between one another in order to compute the outcome of the secure GE protocol.

The implemented network is wired and fully connected using a switch, as illustrated in Figure 6.2.

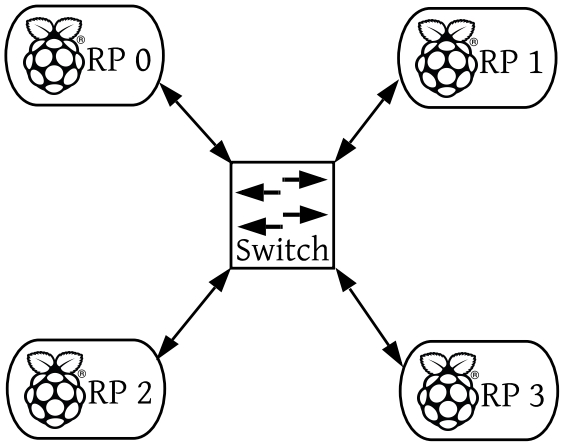


**Figure 6.2:** Network of four Raspberry Pi computers with wired TCP communication using Ethernet cables and a switch.

Note, that cloud computations are obviously based on wireless communication. The purpose of the implementation is to prove the secure GE protocol's feasibility in a real network, thus wired communication is reasonable to utilize.

The communication chosen to be utilized is the connection oriented Transmission Control Protocol (TCP). It is important that all transmitted data is received, as

$\text{MPC}_{\text{omp}}$ relies on a majority of shares to compute correct outputs, which TCP guarantees [Tanenbaum and Wetherall, 2011, p. 47]. Note, that missing some shares are not necessarily problematic, as the Shamir's SSS only requires $t + 1$ shares to be able to reconstruct a secret. This suggests that a connectionless protocol can be successfully used to achieve correct $\text{MPC}_{\text{omp}}$ results, even if some shares are lost in the communication. A connectionless protocol is, however, not chosen as the amount of missing shares, tolerable within Shamir's SSS, is preserved to handle corrupt parties rather than communication errors.

TCP is based on sending acknowledgements when data has been successfully received. This introduces transit delays [Tanenbaum and Wetherall, 2011, p. 36] and can become extensive for $\text{MPC}_{\text{omp}}$ as the secure computations are many compared to open computing. The computational cost of the secure GE protocol in relation to open Gaussian elimination is presented and discussed in chapter 7.

The following section presents the network setup from a spectator's perspective.

## 6.2 Setup Presentation

In order to visualize the interaction between the cloud servers and the data owner, each party displays different operations while executing the secure GE protocol.

Figure 6.3 displays the data owner's RP interface.

```
Data owner, autonomous vehicle

Connection established

Input matrix A:
[[2 3]
 [4 9]]
Observation vector b:
[[ 6]
 [15]]
Preconditions satisfied: the system is solvable

Shares have been send to cloud servers
...
```

**Figure 6.3:** Depiction of the display of the RP that constitutes the data owner. The data owner distributes shares to the cloud servers.

First a connection must be successfully established between all parties. The linear system that must be securely solved is then printed such that the spectator can cross-check the impending solution. According to chapter 4, preconditions must be satisfied to ensure that the linear system is solvable and that the secure GE protocol is applicable. In this case the preconditions are satisfied and it is stated on the data owner's display. The data owner can therefore proceed and thus creates and distributes shares to the cloud servers.

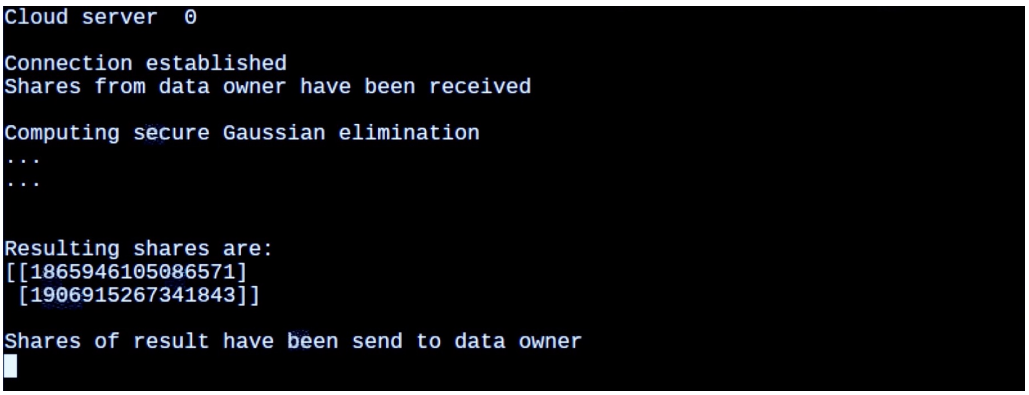


**Figure 6.4:** Depiction of the display of one of the three RPs that constitutes a cloud server. The cloud servers compute the secure GE protocol.

One of the three cloud servers' RP display is depicted in Figure 6.4. First the cloud server is identified. Then connection is established, similar to the data owner. Each cloud then receives shares from the data owner, from which the cloud servers compute the secure GE protocol. For the sake of providing insights to the spectator the shares of the computed result are displayed. Note, that the shares are private and must not be broadcast as anyone will then be able to reconstruct the private result. The shares of the computed result are lastly send to the data owner.


**Figure 6.5:** Depiction of the display of the RP that constitutes the data owner. The data owner reconstructs the result.

The data owner reconstructs the private result and displays it for the purpose of allowing the spectator to cross-check the solution, see Figure 6.5.

The secure GE protocol has been successfully implemented in a network consisting of one data owner and three cloud servers. The code can be accessed on: https://github.com/avtl/thesis/RPnetwork.

As the simulation implementation and the RP network implementation of the secure GE protocol have been described, the results are presented and discussed in the following chapter.

# 7 | Results

This chapter presents the results of the secure GE protocol for solving linear systems in order to solve unconstrained and equality constrained $\text{MPC}_{\text{ontrol}}$ problems.
In chapter 5 it was concluded that the inversion method used within the secure GE protocol, see Protocol 4.2, introduces precision loss. The output of the secure GE protocol is consequently approximate solutions of linear systems. The precision loss and what causes it are firstly presented, followed by an evaluation of the computational cost of secure computations in relation to real time control feasibility.

## 7.1 Precision

This section presents the results of multi thread simulations of the secure GE protocol. First a description of the variables that influence the precision is presented followed by simulations of one linear system with different simulation settings. Based on the simulation results the security and applicability of secure GE protocol is discussed. Two additional linear system are afterwards simulated to examine the influence the linear system may have in relation to the secure GE protocol's applicability.

### 7.1.1 Influencing Variables

The leading principal minor of $[\![\mathbf{A}']\!] = \mathbf{U}[\![\mathbf{A}]\!]\mathbf{L}$ and the rank of $[\![\mathbf{A}]\!] \,|\, [\![\mathbf{b}]\!]$ must satisfy the preconditions of the secure GE protocol to ensure its applicability, see chapter 4. The Toeplitz matrices $\mathbf{U}$ and $\mathbf{L}$ are generated with independent uniformly distributed random variables within the finite field $\mathbb{F}_p$. The secure GE protocol performs Gaussian elimination on the modified system of $[\![\mathbf{A}']\!] = \mathbf{U}[\![\mathbf{A}]\!]\mathbf{L}$ and $[\![\mathbf{b}']\!] = \mathbf{U}[\![\mathbf{b}]\!]$ as it has a probability greater than $1 - \mu(\mu + 1)/p$ of satisfying the leading principal minor precondition.

The precision loss in the secure GE protocol's output is caused by an insufficient inversion method, see Protocol 5.1, that is invoked before the system modification in the secure GE protocol is revoked. This means that the contributions of the Toeplitz matrices are not entirely revoked, thus the range of the random independent uniformly distributed Toeplitz entries affects the output. It is anticipated that the larger the entries are, the larger the precision loss will be, as the lack of revocation will be more dominating due to the multiplication of larger entries. The insufficient inversion method relies on a random variable, hereafter referred to as the inversion variable, which also affects the output. The inversion variable is multiplied by the private value that must be inverted. The product is scaled by a fixed value smaller than $p-1$ to avoid wrap-around and faulty outputs. The scaling introduces precision loss and as the scaling is the fixed, the larger the inversion variable is the larger the

precision loss will be.
The influence of both the Toeplitz matrices and the inversion variable are examined in the following section.

### 7.1.2 One Linear System with Different Simulation Settings

One linear system is simulated with four different simulation settings to evaluate the influence, that the ranges of the Toeplitz entries and the inversion variable have on the precision.
Note, that all simulations are done using the cardinality prime $p = 792606555396977$. All simulation settings are simulated 1000 times.

The linear system is:

$$[\![\mathbf{A}]\!] = \begin{bmatrix} [\![2]\!] & [\![3]\!] \\ [\![4]\!] & [\![9]\!] \end{bmatrix}, \qquad [\![\mathbf{b}]\!] = \begin{bmatrix} [\![6]\!] \\ [\![15]\!] \end{bmatrix}, \qquad \text{solution: } [\![\mathbf{x}]\!] = \begin{bmatrix} [\![1.5]\!] \\ [\![1]\!] \end{bmatrix}. \quad (7.1)$$

The Toeplitz entries and the inversion variable are initially generated as random integers in the small range [1; 9] as listed in Table 7.1.

| Computations | Toeplitz entries | Inversion variable |
|---|---|---|
| 1000 | [1; 9] | [1; 9] |

**Table 7.1:** Number of computations and the ranges that the Toeplitz entries and the inversion variable are randomely generated within.

The percentage error deviations of the approximate solutions are determined as:

$$e_i = \frac{X_i - x_i}{X_i} \cdot 100 \,, \quad (7.2)$$

where:

$X_i$ is the true solution for $i \in \{1, 2\}$, see Equation 7.1,

$x_i$ is the approximate solution for $i \in \{1, 2\}$.

The distributions of the percentage error deviations $e_1$ and $e_2$ can be seen in Figure 7.1 and Figure 7.2, respectively. The figures illustrate the lower and upper bounds that outline the interval that the secure GE protocol outcome with 95 % probability will fall within.

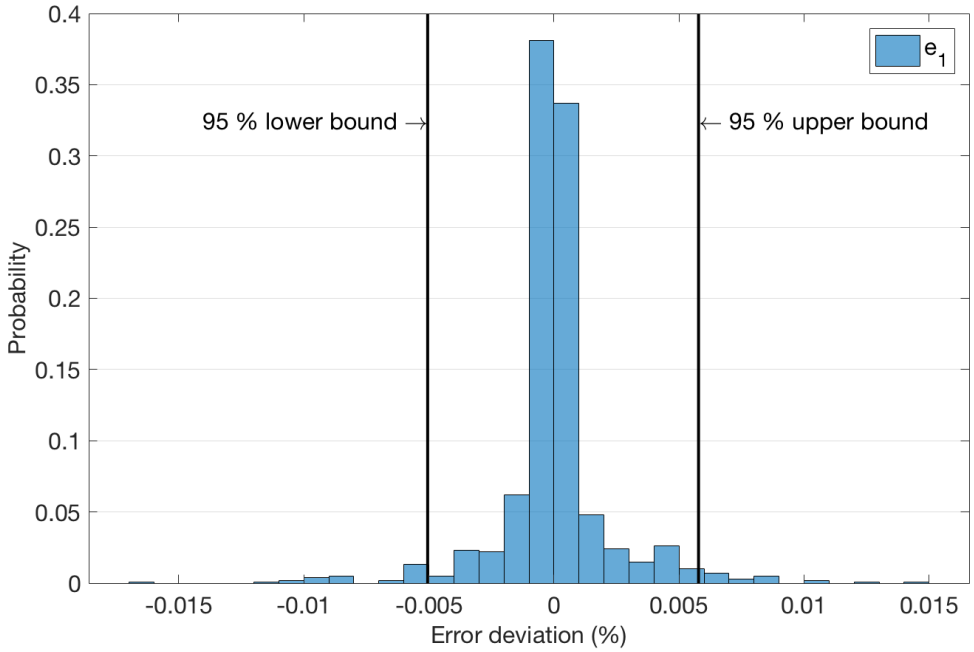


**Figure 7.1:** Distribution of the percentage error deviation $e_1$ based on the simulation setting listed in Table 7.1.

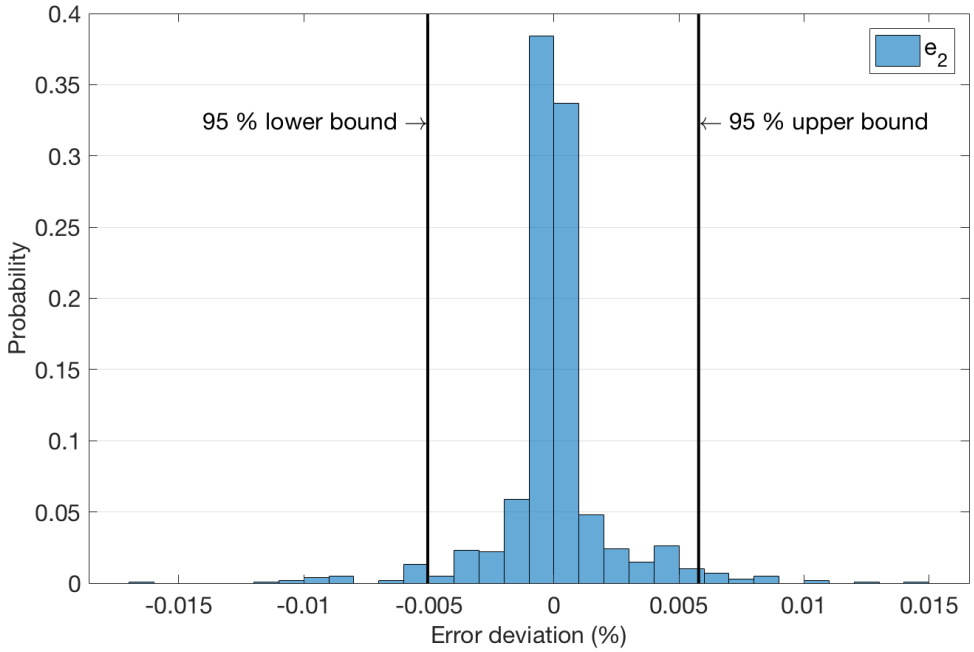


**Figure 7.2:** Distribution of the percentage error deviation $e_2$ based on the simulation setting listed in Table 7.1.

First, notice how the distributions are almost identical. This is because the secure GE protocol computes both solutions of the two dimensional linear system using the same Toeplitz matrices and the same inversion variable, as it utilizes Gaussian elimination.

The 95 % lower and upper bounds for the data sets are almost symmetric and approximately $\pm 0.005$ %. This means that 5 % of the computed outcomes will deviate more than $\pm 0.005$ % of the true solution $X_i$.
The standard deviations are:

$$\sigma(e_1) = 2.275 \cdot 10^{-3} ,$$
$$\sigma(e_2) = 2.275 \cdot 10^{-3} .$$

As the distributions are almost identical the standard deviations are also approximately identical.
The level of tolerable precision loss depends on the control application the solutions must be used within. The presented values are a relative measure used to gain knowledge of how the ranges of the Toeplitz entries and the inversion variable influence the precision.
The precision loss of the presented results is very small as anticipated, as the Toeplitz entries and the inversion variable have been generated randomly within the small range [1; 9]. It is interesting to examine how larger ranges affect the precision.

First the range of the Toeplitz entries is increased to [1; 50] followed by increasing the range of the random inversion variable to [1; 50] as seen in Table 7.2.

| Setting | Computations | Toeplitz entries | Inversion variable |
|---|---|---|---|
| $a$ | 1000 | [1; 50] | [1; 9] |
| $b$ | 1000 | [1; 9] | [1; 50] |

**Table 7.2:** Setting ID, number of computations and the ranges that the Toeplitz entries and the inversion variable are randomely generated within.

As the distributions for the initial simulation setting in Table 7.1 are almost identical, see Figure 7.1 and Figure 7.2, it is chosen to only present the distribution of $e_1$ for each simulation setting. Note, that all distributions can be accessed on:
https://github.com/avtl/thesis/MATLAB.

The distributions of the two different simulation settings $a$ and $b$ are seen in Figure 7.3 and Figure 7.4, respectively.

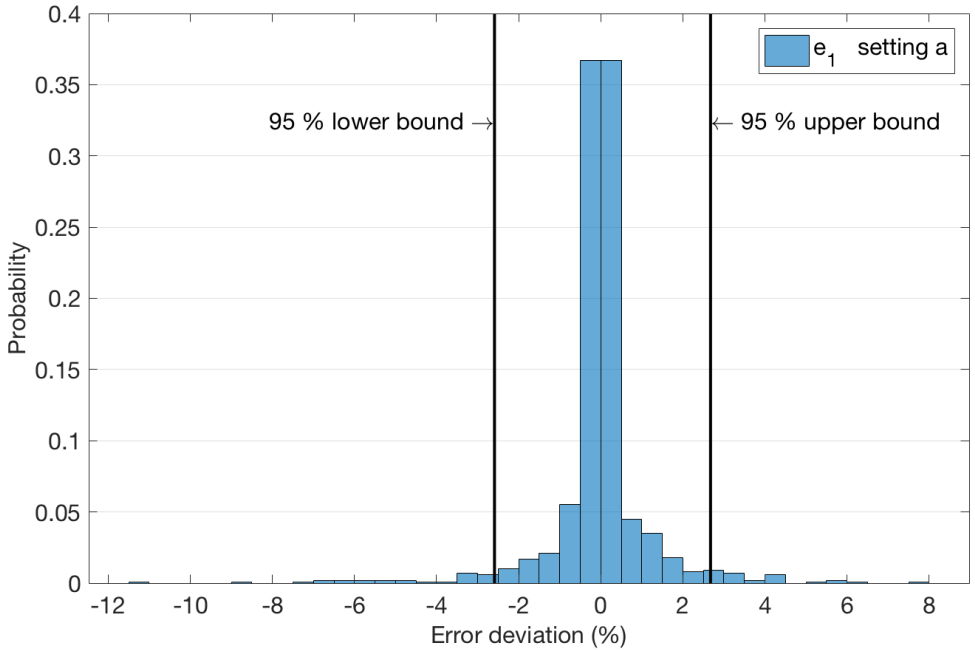


**Figure 7.3:** Distribution of $e_1$ based on simulation setting $a$ from Table 7.2.

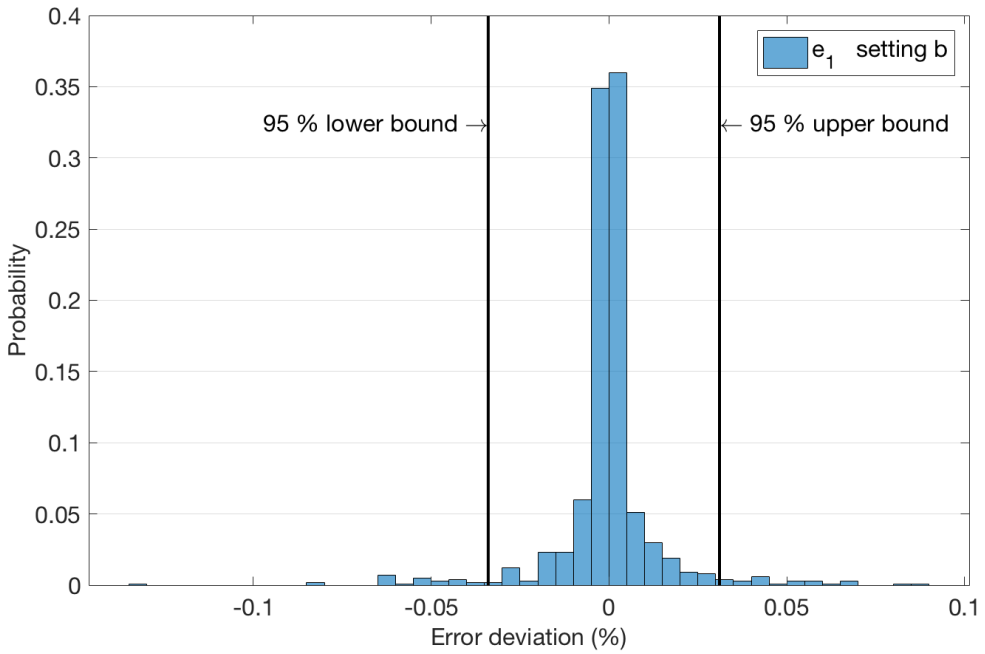


**Figure 7.4:** Distribution of $e_1$ based on simulation setting $b$ from Table 7.2.

The distribution of setting $a$ in Figure 7.3 has a large percentage error deviation compared to the distribution of setting $b$ in Figure 7.4. The simulation setting $a$, where the Toeplitz entries are random variables in the range [1;50], yields a large precision loss. The 95 % of the samples lie within the range [-2.588 %; 2.675 %] compared to the much smaller range [-0.034 %; 0.031 %] of simulation setting $b$.

The standard deviations are:

$$\sigma(a_{e_1}) = 1.245 \ ,$$
$$\sigma(b_{e_1}) = 0.015 \ .$$

The standard deviations also affirm the dominant influence that the increased range of the Toeplitz entries has on the precision. The standard deviation of setting $a$ is more than 80 times lager than the standard deviation of setting $b$. This is because both the Toeplitz matrices $\mathbf{U}$ and $\mathbf{L}$ contribute to the modified input matrix $[\![\mathbf{A}']\!] = \mathbf{U}[\![\mathbf{A}]\!]\mathbf{L}$. Their influence is therefore expected to be larger than the inversion variable that is only once multiplied with a single variable.

How the ranges of the random variables relate to the secure GE protocol's security and applicability is considered in the following section.

### 7.1.3 Security and Applicability

It has been established that the ranges of the Toeplitz entries and the inversion variable influence the precision of the secure GE protocol's output. Increasing the range of the random Toeplitz entries affects the precision loss immensely compared to increasing the range of the inversion variable.
The precision loss occurs due to an insufficient inversion method that is invoked in the secure GE protocol.

The range of the random inversion variable influences the level of security as the inversion variable is multiplied by private information that is then broadcast. If the interval is small it is more likely that over time tendencies can be intercepted and private information may be disclosed.

The Toeplitz matrices are not secret, thus the range of the Toeplitz entries do not influence the level of security. The Toeplitz matrices, however, are used to increase the probability that the secure GE protocol is applicable and hence capable of producing a solution. The probability is greater than $1 - \mu(\mu + 1)/p$, according to [Bouman and Vreede, 2018], if the Toeplitz entries are generated with random independent uniformly distributed variables within the finite field with cardinality prime $p$.
The previous section simulated the secure GE protocol using Toeplitz entries ranging from [1; 9] and [1; 50] instead of [1; $p - 1$]. This should certainly decrease the probability of satisfying the aforementioned precondition for the secure GE protocol.

The variable $\mu$ is the order of the square input matrix $[\![\mathbf{A}]\!]$, which is 2 in this case. This suggests that variables [1; 6] must yield a fail precondition of $[\![\mathbf{A}']\!]$. The linear system in Equation 7.1 has been simulated 2000 times in total using the Toeplitz entries range [1; 9]. According to the probability stated in [Bouman and Vreede, 2018], it is reasonable to expect approximately 2/3 of the simulations to fail. However, no simulation failed. Neither did any of the 1000 simulations using the [1; 50] range for the Toeplitz entries. This indicates that the lower bounds are not tight. It may, however, be the linear system that has favourable conditions, thus different linear systems must be considered. This is done in a later section.

Until an adequate method for secure inversion is developed, it is not justifiable to use Toeplitz entries generated with random variables from a large range due to disruptive precision loss. It is interesting to examine if the [1; 9] range of the Toeplitz entries will produce failed outputs when utilized with other linear systems. Before investigate other linear systems, the system in Equation 7.1 is lastly simulated using the ranges seen in Table 7.3, which constitute a compromise between precision and security.

| Setting | Computations | Toeplitz entries | Inversion variable |
|---|---|---|---|
| $c$ | 1000 | [1; 9] | [1; 500] |

**Table 7.3:** Number of computations and the ranges that the Toeplitz entries and inversion variable are randomely generated within.

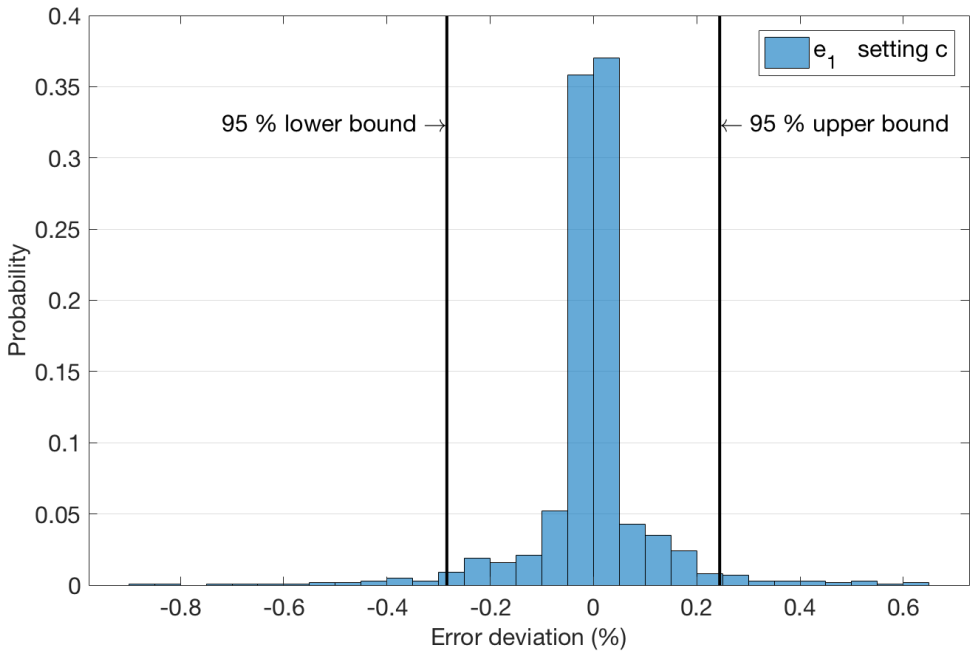


**Figure 7.5:** Distribution of $e_1$ based on simulation setting $c$ from Table 7.3.

The standard deviation of the distribution in Figure 7.5 is:

$$\sigma(c_{e_1}) = 0.121 \ .$$

It can be seen that 95 % of the data lie within [-0.284 %; 0.244 %] error deviation of the true solution $X_1$ and has a standard deviation of approximately 0.12. It is moreover relevant to notice, that no simulation result is faulty despite the small ranged Toeplitz entries.
The simulation setting $c$ is with no further inquiry utilized to simulate different linear systems.

### 7.1.4 Different Linear Systems

Two additional linear systems are considered to investigate the applicability of the secure GE protocol using small ranged Toeplitz entries.

System 1 is given as:

$$[\![\mathbf{A}]\!] = \begin{bmatrix} [\![5]\!] & [\![2]\!] \\ [\![6]\!] & [\![3]\!] \end{bmatrix}, \qquad [\![\mathbf{b}]\!] = \begin{bmatrix} [\![2]\!] \\ [\![12]\!] \end{bmatrix}, \qquad \text{solution: } [\![\mathbf{x}]\!] = \begin{bmatrix} [\![-6]\!] \\ [\![16]\!] \end{bmatrix}. \quad (7.3)$$

System 2 is given as:

$$[\![\mathbf{A}]\!] = \begin{bmatrix} [\![2]\!] & [\![2]\!] \\ [\![-3]\!] & [\![2]\!] \end{bmatrix}, \qquad [\![\mathbf{b}]\!] = \begin{bmatrix} [\![12]\!] \\ [\![-3]\!] \end{bmatrix}, \qquad \text{solution: } [\![\mathbf{x}]\!] = \begin{bmatrix} [\![3]\!] \\ [\![3]\!] \end{bmatrix}. \quad (7.4)$$

Note, that the systems have been chosen such that system 1's solutions include a negative value and that system 2's input matrix and observation vector include negative values. This is done to ensure that the secure GE protocol is applicable for linear systems independent on the existence of negative values without distinction of where these entries occur in the linear systems.

The distributions of system 1 and system 2 with the final simulation setting $c$ is seen in Figure 7.6 and Figure 7.7, respectively.

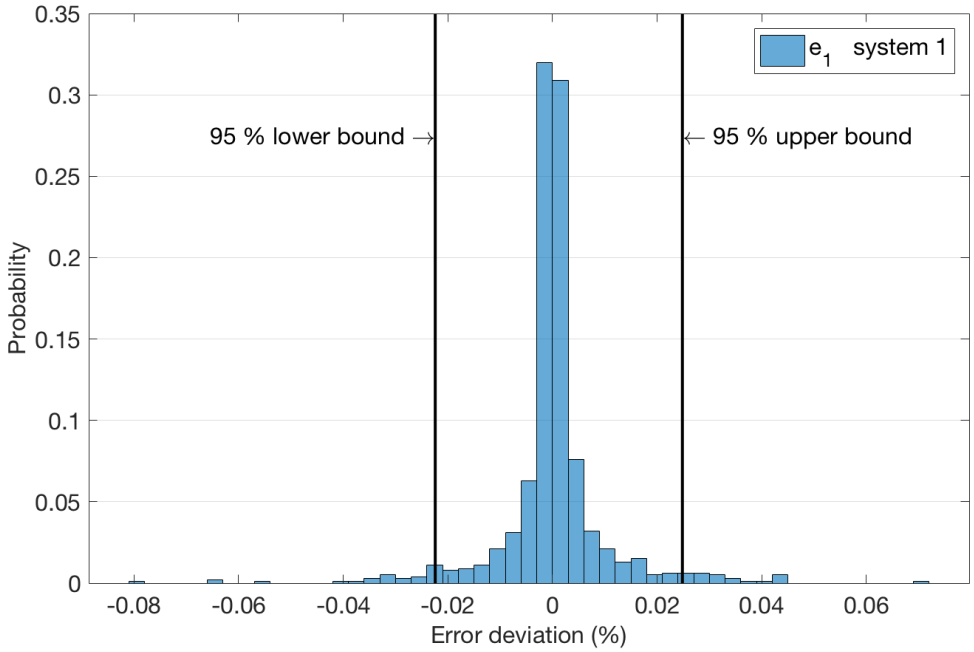


**Figure 7.6:** Distribution of $e_1$ of sytem 1 based on simulation setting $c$ from Table 7.3.

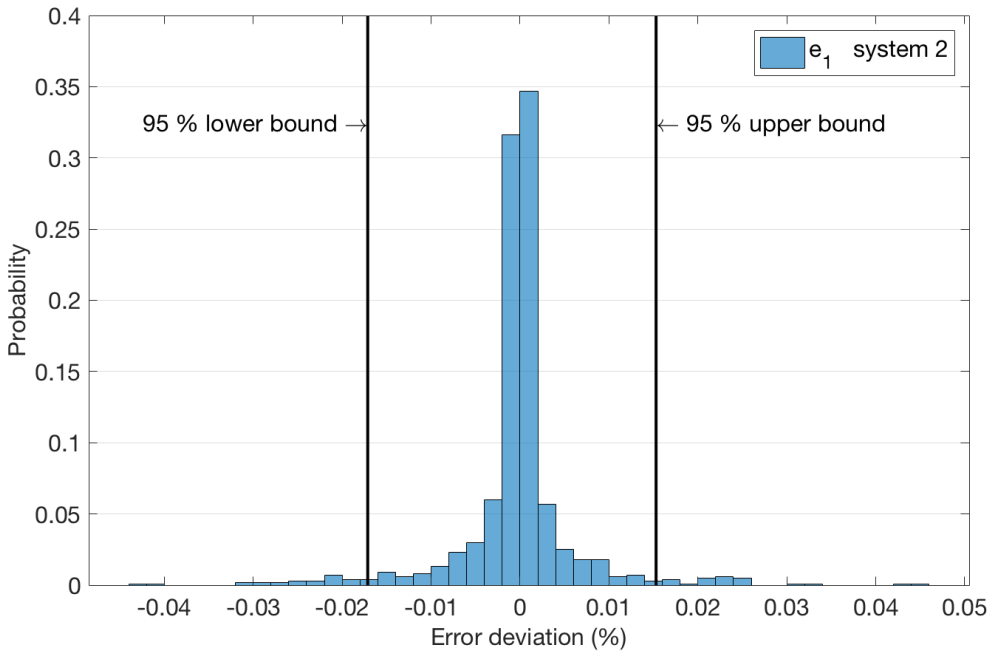


**Figure 7.7:** Distribution of $e_1$ of system 2 based on simulation setting $c$ from Table 7.3.

The 95 % lower and upper bounds are:

$$\begin{aligned} &\text{System 1:} \quad [-0.022\ \%;\ 0.024\ \%]\ , \\ &\text{System 2:} \quad [-0.017\ \%;\ 0.015\ \%]\ . \end{aligned}$$

The standard deviations are:

$$\begin{aligned} &\text{System 1:} \quad \sigma(e_1) = 10.215 \cdot 10^{-3}\ , \\ &\text{System 2:} \quad \sigma(e_1) = 6.879 \cdot 10^{-3}\ . \end{aligned}$$

Based on the distributions and the standard deviations it can be seen that the small ranged Toeplitz entries do not produce faulty outputs. It can be deduced that the probability for successful application of the secure GE protocol is not a tight bound as the probability has proven to be much greater than $1 - \mu(\mu + 1)/p$. The paper [Bouman and Vreede, 2018] states that the probability is greater than $1-\mu(\mu+1)/p$, which is certainly true, though it can be further restricted to present a more reliable probability bound.

The secure GE protocol for solving linear systems of both positive and negative values has demonstrated to be applicable.

The precision loss of the secure GE protocol and what causes it have been presented and discussed. Now follows considerations on the computational cost of the secure GE protocol.

## 7.2 Computational Cost

It is essential to consider the computational cost of the secure GE protocol to address its feasibility within real time control applications. $\text{MPC}_{\text{omp}}$ performs many additional computations compared to open computing. Many computations within the secure GE protocol require shares to be distributed between the collaborating parties to preserve privacy.

It is chosen to compare the computation time of regular open Gaussian elimination and the secure GE protocol.

### 7.2.1 Computation Time

Computation time is a relative measure as it depends on the specifications of the machines that compute the open and secure results. Moreover simultaneous activities may influence the computational speed, as the processing power is distributed among the running activities. It is therefore important that the computation times are obtained by the same machines with no unnecessary activities running while timing the open and the secure computations. Comparing computation times provides a

reasonable indication of what the additional computational cost is for solving linear systems securely.

The secure GE protocol is computed within the implemented RP network rather than as a multiple thread simulation. The RP network includes communication latency different from the simulation. Communication latency is vital to consider in regards to the feasibility of $\text{MPC}_{\text{omp}}$ in real time control.
To ensure comparability of the measured computation times, the open Gaussian elimination must also be computed on the RPs.
All computations are done in Python, such that different programming languages, and thus different kernels, do not influence the comparison differently. It is moreover the same linear system, see Equation 7.1, that the open and secure Gaussian elimination must solve to ensure a fair comparison.

Note, that the secure GE protocol utilizes *Beaver's triplets* for secure multiplication. Generating these and distributing them among the collaborating parties are done in a preprocessing phase. The preprocessing phase can be executed when there is no need for efficient $\text{MPC}_{\text{omp}}$ e.g. when an autonomous vehicle is parked, thus it is not included in the computation time of the secure GE protocol.

As computation time is influenced by simultaneous activities within the computing machine, all computations will be performed 10 times and the average will be used for comparison.

The average computation times for open Gaussian elimination for the individual RPs are:

| | |
|---|---|
| RP 0, cloud server: | $2.061 \cdot 10^{-3}$ seconds |
| PR 1, cloud server: | $2.071 \cdot 10^{-3}$ seconds |
| RP 2, cloud server: | $2.053 \cdot 10^{-3}$ seconds |
| RP 3, autonomous vehicle: | $2.227 \cdot 10^{-3}$ seconds |

The open Gaussian elimination computations are run on each RP as the computation time of the secure GE protocol depends on the computational speed of all four RPs. If one RP is delaying the computation of the secure GE protocol it must be considered in the comparison. The computation times of the four RPs are therefore averaged to most reliably represent the computation time for open Gaussian elimination.

The average computation time of open Gaussian elimination for the four RPs is:

RPs average: $2.114 \cdot 10^{-3}$ seconds

Timing the secure GE protocol requires the programming script for each party to be started manually. The cloud servers are started first as the secure GE protocol and the timing first start when the data owner is started.

The average computation time of the secure GE protocol is:

Secure GE protocol: 3.092 seconds

The ratio between the computation time of the open and the secure Gaussian elimination is:

$$\frac{3.092 \text{ s}}{2.114 \cdot 10^{-3} \text{ s}} \approx 1463 \tag{7.5}$$

This is immensely expensive, especially when considering the fact that the communication is wired and does not include delays caused by complex encryption, packet loss and retransmissions. This will be the case in real life implementation and must be expected to have even greater computational costs than what has been seen in the wired RP network. Considerations on computational costs are presented in the following.

### 7.2.2 Considerations on Computational Costs

It is not surprising that the computation time for solving linear systems using the secure GE protocol is greater than by open Gaussian elimination. $\text{MPC}_{\text{omp}}$ performs many additional computations that depends on the distribution of shares between parties.

The computation time ratio is, however, not satisfying seen from a real time control perspective, as the control inputs will simply not be computed fast enough to obtain any desired performance in most applications.
Computational efficiency has not been a focus in this thesis, however, the *Beaver's triplet* multiplication protocol with improved efficiency has been utilized.
Research in optimizing $\text{MPC}_{\text{omp}}$ efficiency is a research topic in itself. The reader is referred to [Polychroniadou, 2016] if one is interested in gaining knowledge within this line of research.

Before real time control can be realized securely in the cloud, further research into $\text{MPC}_{\text{omp}}$ efficiency within a control context must be conducted.

This finalizes the presentation and discussion of the results. The thesis conclusion is presented next.

# 8 | Conclusion

The focus of this thesis has been to research how control can be securely computed in the cloud. The unconstrained and equality constrained $\text{MPC}_{\text{ontrol}}$ problems can take form as linear systems that can be solved using Gaussian elimination. $\text{MPC}_{\text{omp}}$ using secret sharing was investigated to utilize secure Gaussian elimination as a solution to the control problems of interest.
The secure GE protocol was simulated and the simulation results were considered and discussed. Due to an insufficient inversion method, precision loss was introduced and affected the solutions computed by the secure GE protocol.
The secure GE protocol was moreover successfully implemented on a network of four RPs. The computation time for the implemented secure GE protocol was compared with the average computation time of open Gaussian elimination of the RPs. The computational cost for solving linear systems securely compared with open computations was approximately 1463:1. This ratio is infeasible in a real time control context, especially as it does not include all aspects of communication latency as the RP network was wired.
It is therefore vital that efficiency of $\text{MPC}_{\text{omp}}$ is considered and optimized before it can be used to securely solve real time control problems.

The thesis constitutes initial research into the field of secure control computations using $\text{MPC}_{\text{omp}}$. This field shows immense potential and should be investigated further.

# 9 | Future Work

This chapter presents perspectives for future research as a continuation of this work.

- In chapter 5 it was established that the implemented inversion method, which the secure GE protocol invokes, is insufficient in providing exact inverting. It operates within multiple fields where a small scaling factor is required to ensure injective mapping. The fixed scaling introduces precision loss of the inverted variable and the precision loss affects the output of the secure GE protocol. Further research into secure inversion methods within finite field arithmetic must be conducted in order to realise the full potential of the secure GE protocol.
- The focus of the thesis was delimited to only consider the unconstrained and equality constrained $\text{MPC}_{\text{ontrol}}$ problems. They are both theoretical problems. No systems can exist without constraints and equality constrained problems are prone to be time variant, hence switching between a set of equality constrained controllers is necessary. Switching in secure $\text{MPC}_{\text{omp}}$ must be investigated. It requires comparison which the *SECURE* project is currently researching. Moreover the inequality constrained $\text{MPC}_{\text{ontrol}}$ problem must be solved securely. This is a non trivial extension of this work as it is currently solved using software solvers in the industry. These methods are hard to convert to secure $\text{MPC}_{\text{omp}}$ protocols. If future research accomplishes to solve inequality constrained $\text{MPC}_{\text{ontrol}}$ it will be ground breaking for the industry and its security.
- In any control problem solved securely using $\text{MPC}_{\text{omp}}$, latency must be studied as it is problematic in real time control problems with considerably fast dynamics. Control of an autonomous vehicle requires immensely fast computations. Adding latency due to $\text{MPC}_{\text{omp}}$ and encrypted wireless communication to the secure control challenge poses an even greater challenge. Moreover data packets must be expected to be lost occasionally thus this must be accounted for in the secure solution. These perspectives must be researched in order to eventually achieve secure control computations in the cloud for systems with fast dynamics.

Further research must be conducted within this field prior to become suitable for real life control problems. It has, however, demonstrated great potential to become epochal in future control applications where security is a priority.

# Bibliography

AAU Strategy (2017). Secure estimation and control using recursion and encryption (secure). https://www.strategi.aau.dk/Forskning/Tv%C3%A6rvidenskabelige+forskningsprojekter/SECURE/. [Online; accessed 23-February-2019].

AAU Strategy (2018). Tværvidenskabelige forskningsprojekter udvalgt. https://www.strategi.aau.dk/aktuelt/nyhed/tvaervidenskabelige-forskningsprojekter-udvalgt.cid336771. [Online; accessed 23-February-2019].

AAU TANT (2018). Project: Secure estimation and control using recursion and encryption (secure). https://www.tant.aau.dk/energy-environment-sustainability-future-making/Projects/project-secure/. [Online; accessed 23-February-2019].

Ahkâm (2017). Icon vector hacker. https://www.freeiconspng.com/img/37219. [Online; accessed 18-May-2019].

Bar-Ilan, J. and Beaver, D. (1989). Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 201–209. ACM.

Bouman, N. J. and Vreede, N. d. (2018). *New Protocols for Secure Linear Algebra: Pivoting-Free Elimination and Fast Block-Recursive Matrix Decomposition*. Technische Universiteit Eindhoven the Netherlands.

Cramer, R., Damgård, I. B., and Nielsen, J. B. (2015). *Secure multiparty computation*. Cambridge University Press.

Hull, R. (2017). Photo in Introduction. https://www.thisismoney.co.uk/money/cars/article-4095614/Driverless-cars-increase-congestion-UK-roads.html. [Online; accessed 27-February-2019].

Kaltofen, E. and Saunders, D. (1991). On wiedemann's method of solving sparse linear systems.

Maciejowski, J. M. (2000). *Predictive control: with constraints*. Prentice Hall, an imprint of Pearson Education.

Nishide, T. and Ohta, K. (2007). Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *International Workshop on Public Key Cryptography*, pages 343–360. Springer.

Polychroniadou, A. (2016). *On the Communication and Round Complexity of Secure Computation*. Aarhus University.

SimplyScience, Y. (2018). Basics of secure multiparty computation. https://www.youtube.com/watch?v=_mDlLKgiFDY&t=590s. [Online; accessed 15-February-2019].

Tanenbaum, A. S. and Wetherall, D. J. (2011). *Computer Networks, 5th edition*. Pearson Education.

Tjell, K. S. (2018). *Privacy Preserving Control Using Multiparty Computation*. Aalborg University.

Weisstein, E. W. (2018). Pivoting. http://mathworld.wolfram.com/Pivoting.html. [Online; accessed 2-May-2019].

# A | Protocol 2.2 Example

An example of secure multiplication using Protocol 2.2 from chapter 2 is presented, as finite field arithmetic can perhaps seem a bit confusing if one is not familiar.

Number of parties: $n = 3$.
Polynomial degree: $t = 1$.
Finite field cardinality prime: $p = 7$.
This example multiplies $[\![a; f_a]\!]$ and $[\![b; f_b]\!]$. The secrets are $a = 2$ and $b = 3$.

Recall that according to the polynomial definition of Shamir's SSS a first degree polynomial is defined as:

$$f_s(x) = s + c_1 x \ , \tag{A.1}$$

where $s$ is the secret and $c_1$ is a random uniformly distributed variable.

In this example the polynomials are:

$$f_a(x) = 2 + x \ , \tag{A.2}$$
$$f_b(x) = 3 + 2x \ . \tag{A.3}$$

As the calculations are done within the finite field $\mathbb{F}_7$ the shares must be calculated using modulo 7. The parties hold the shares:

| | $f_a(x)$ | $f_b(x)$ |
|---|---|---|
| $P_1$ | $2 + 1 \bmod 7 = 3$ | $3 + 2 \cdot 1 \bmod 7 = 5$ |
| $P_2$ | $2 + 2 \bmod 7 = 4$ | $3 + 2 \cdot 2 \bmod 7 = 0$ |
| $P_3$ | $2 + 3 \bmod 7 = 5$ | $3 + 2 \cdot 3 \bmod 7 = 2$ |

Consider $P_3$'s share of $f_b$ for instance. It is calculated as follows:

$$f_b(3) = 3 + 2 \cdot 3 = 9 \ , \tag{A.4}$$
$$9 \bmod 7 = 2 \ . \tag{A.5}$$

In finite field arithmetic 9 divided by 7 leaves a quotient of 1 and a remainder of 2, thus the result is 2.

With the basics covered the secure multiplication protocol is now presented.

## Protocol 2.2

**Step 1:** The parties compute

$$[\![ab; h]\!]_{2t} = [\![a; f_a]\!]_t * [\![b; f_b]\!]_t \ , \tag{A.6}$$

where $h = f_a f_b$ that is a $2t$ degree polynomial. Note, that $h(0) = ab$.

$P_i$ multiplies the shares $a_i$ and $b_i$, as follows:

$$P_1 : \ h(1) = \ f_a(1) \cdot f_b(1) = 3 \cdot 5 \bmod 7 = 1 \ , \tag{A.7}$$
$$P_2 : \ h(2) = \ f_a(2) \cdot f_b(2) = 4 \cdot 0 \bmod 7 = 0 \ , \tag{A.8}$$
$$P_3 : \ h(3) = \ f_a(3) \cdot f_b(3) = 5 \cdot 2 \bmod 7 = 3 \ . \tag{A.9}$$

**Step 2:** $P_i$ distributes $[\![h(i); f_i]\!]_t$.

Note, that $f_i(x)$ where $i$ =1,2,3 is defined as:

$$f_i(x) = h(i) + dx \ , \tag{A.10}$$

where d is a random uniformly distributed variable according to the polynomial definition of Shamir's SSS.

In this example $P_i$ creates $f_i(x)$ as:

$$P_1 : \ f_1(x) = 1 + 1x \ , \tag{A.11}$$
$$P_2 : \ f_2(x) = 0 + 3x \ , \tag{A.12}$$
$$P_3 : \ f_3(x) = 3 + 1x \ . \tag{A.13}$$

Party $P_i$ distributes its shares calculated from $f_i(x)$, such that the parties hold:

| | $f_1(x)$ | $f_2(x)$ | $f_3(x)$ |
|---|---|---|---|
| $P_1$ | $f_1(1)$ | $f_2(1)$ | $f_3(1)$ |
| $P_2$ | $f_1(2)$ | $f_2(2)$ | $f_3(2)$ |
| $P_3$ | $f_1(3)$ | $f_2(3)$ | $f_3(3)$ |

In this example the shares are as follows:

| | $f_1(x)$ | $f_2(x)$ | $f_3(x)$ |
|---|---|---|---|
| $P_1$ | $1 + 1 \cdot 1 \bmod 7 = 2$ | $0 + 3 \cdot 1 \bmod 7 = 3$ | $3 + 1 \cdot 1 \bmod 7 = 4$ |
| $P_2$ | $1 + 1 \cdot 2 \bmod 7 = 3$ | $0 + 3 \cdot 2 \bmod 7 = 6$ | $3 + 1 \cdot 2 \bmod 7 = 5$ |
| $P_3$ | $1 + 1 \cdot 3 \bmod 7 = 4$ | $0 + 3 \cdot 3 \bmod 7 = 2$ | $3 + 1 \cdot 3 \bmod 7 = 6$ |

**Step 3:** The parties compute

$$[\![\sum_{i=1}^{n} r_i h(i); \sum_{i=1}^{n} r_i f_i]\!]_t \ . \tag{A.14}$$

The recombination vector is used in order to reconstruct the secret result hidden in the shares. Recall that each entry of the recombination vector is defined as:

$$r_i = \delta_i(0) = \prod_{j \in C, j \neq i} \frac{-j}{i-j} \ , \tag{A.15}$$

where $C$ is the set of $n$ indices of the shares.

The entries of the recombination vector are:

$$r_1 = \left(\frac{-2}{1-2} \cdot \frac{-3}{1-3}\right) \mod 7 = 2 \cdot 5 \mod 7 = 3 \ , \tag{A.16}$$

$$r_2 = \left(\frac{-1}{2-1} \cdot \frac{-3}{2-3}\right) \mod 7 = 6 \cdot 3 \mod 7 = 4 \ , \tag{A.17}$$

$$r_3 = \left(\frac{-1}{3-1} \cdot \frac{-2}{3-2}\right) \mod 7 = 3 \cdot 5 \mod 7 = 1 \ . \tag{A.18}$$

To support the unfamiliar reader the approach for solving Equation A.16-A.18 is explained step by step.

Consider $r_3$ in Equation A.18 where the first fraction is -1/2. One may immediately think that it equals -0.5. This is not the case in finite field arithmetic. Instead the fraction must be expressed according to finite field arithmetic, as:

$$\frac{-1}{2} \mod 7 = \frac{6}{2} \ . \tag{A.19}$$

An intuitive way of thinking about modular division is by considering the inverse, such that:

$$\frac{6}{2} \mod 7 = 6 \cdot \frac{1}{2} \mod 7 \ . \tag{A.20}$$

When a value is multiplied by its inverse it equals one. This is used to determine the finite field arithmetic value of -1/2 in Equation A.20, as:

$$2 \cdot v_{inv} \mod 7 = 1 \ . \tag{A.21}$$

Values to modulo 7 that equals 1 are:

| $7 \cdot 0 + 1 \mod 7$ | $7 \cdot 1 + 1 \mod 7$ | $7 \cdot 2 + 1 \mod 7$ | $7 \cdot 3 + 1 \mod 7$ | $7 \cdot 4 + 1 \mod 7$ |
|---|---|---|---|---|
| 1 | 8 | 15 | 22 | 36 |

**Table A.1:** The first five mod 7 rounds that equal 1.

The inverse of 2 is determined by finding a value in Table A.1 that is divisible by 2. In this case 8 is a multiple of 2 by 4 times. Thus:

$$\frac{1}{2} \bmod 7 = 4 \, . \tag{A.22}$$

The first fraction of $r_3$ in Equation A.18 equals:

$$\frac{-1}{2} \bmod 7 = \frac{6}{2} \bmod 7 = 6 \cdot 4 \bmod 7 = 3 \, . \tag{A.23}$$

Similarly the second fraction of $r_3$ in Equation A.18 is determined as:

$$\frac{-2}{1} \bmod 7 = \frac{5}{1} \bmod 7 = 5 \cdot 1 \bmod 7 = 5 \, . \tag{A.24}$$

The resulting entry of the recombination vector, $r_3$, is thus $3 \cdot 5 \bmod 7 = 1$ as seen in Equation A.18.

According to the protocol $P_i$ calculates the resulting shares of $ab$, as follows:

$$P_1 : \; r_1 \cdot f_1(1) + r_2 \cdot f_2(1) + r_3 \cdot f_3(1) \, , \tag{A.25}$$
$$P_2 : \; r_1 \cdot f_1(2) + r_2 \cdot f_2(2) + r_3 \cdot f_3(2) \, , \tag{A.26}$$
$$P_3 : \; r_1 \cdot f_1(3) + r_2 \cdot f_2(3) + r_3 \cdot f_3(3) \, . \tag{A.27}$$
$$\tag{A.28}$$

In this example the resulting shares are:

$$P_1 : 3 \cdot (1 + 1 \cdot 1) + 4 \cdot (3 \cdot 1) + 1 \cdot (3 + 1 \cdot 1) \bmod 7 = 1 \, , \tag{A.29}$$
$$P_2 : 3 \cdot (1 + 1 \cdot 2) + 4 \cdot (3 \cdot 2) + 1 \cdot (3 + 1 \cdot 2) \bmod 7 = 3 \, , \tag{A.30}$$
$$P_3 : 3 \cdot (1 + 1 \cdot 3) + 4 \cdot (3 \cdot 3) + 1 \cdot (3 + 1 \cdot 3) \bmod 7 = 5 \, . \tag{A.31}$$

Note, that redundant parentheses and multiplication by one appears in Equation A.29-A.31 for the sake of traceability of the polynomial evaluations and the recombination vector's entries to help the reader.

This finalizes Protocol 2.2.

If one is interested in checking if the resulting shares yield the correct product of $a$ and $b$, the recombination vector is utilized, as:

$$P_1 : \; 3 \cdot 1 \bmod 7 = 3 \, , \tag{A.32}$$
$$P_2 : \; 4 \cdot 3 \bmod 7 = 5 \, , \tag{A.33}$$
$$P_3 : \; 1 \cdot 5 \bmod 7 = 5 \, . \tag{A.34}$$

The product $ab$ can be reconstructed by summing the resulting shares that have been multiplied with the recombination vector, see Equation A.32-A.34, as follows:

$$ab = 3 + 5 + 5 \bmod 7 = 6 \, . \tag{A.35}$$

The multiplication result in Equation A.35 is indeed correct.

# B | Secure Matrix Multiplication Example

Let $[\![\mathbf{A}]\!]$ and $[\![\mathbf{B}]\!]$ be two private matrices. Note, that the calculations are done within the finite field with cardinality prime $p = 7$.

$$[\![\mathbf{A}]\!] = \begin{bmatrix} [\![2]\!] & [\![3]\!] \\ [\![1]\!] & [\![2]\!] \end{bmatrix}, \qquad [\![\mathbf{B}]\!] = \begin{bmatrix} [\![3]\!] & [\![1]\!] \\ [\![2]\!] & [\![3]\!] \end{bmatrix}. \tag{B.1}$$

The matrix product yields:

$$[\![\mathbf{AB}]\!] = \begin{bmatrix} [\![5]\!] & [\![4]\!] \\ [\![0]\!] & [\![0]\!] \end{bmatrix}. \tag{B.2}$$

The $[\![\alpha]\!]$ and $[\![\beta]\!]$ are random uniformly distributed and $[\![\gamma]\!] = [\![\alpha]\!][\![\beta]\!]$. The *triplet* can for instance be:

$$[\![\alpha]\!] = \begin{bmatrix} [\![2]\!] & [\![2]\!] \\ [\![1]\!] & [\![3]\!] \end{bmatrix}, \qquad [\![\beta]\!] = \begin{bmatrix} [\![3]\!] & [\![4]\!] \\ [\![2]\!] & [\![1]\!] \end{bmatrix}, \qquad [\![\gamma]\!] = \begin{bmatrix} [\![3]\!] & [\![3]\!] \\ [\![2]\!] & [\![0]\!] \end{bmatrix}. \tag{B.3}$$

Recall from chapter 4, that:

$$[\![\mathbf{AB}]\!] = \mathbf{DE} + \mathbf{D}[\![\beta]\!] + [\![\alpha]\!]\mathbf{E} + [\![\gamma]\!]. \tag{B.4}$$

The matrices $\mathbf{D}$ and $\mathbf{E}$ are broadcast and thus open matrices. They are determined as follows:

$$\mathbf{D} = [\![\mathbf{A}]\!] - [\![\alpha]\!] = \begin{bmatrix} [\![0]\!] & [\![1]\!] \\ [\![0]\!] & [\![6]\!] \end{bmatrix}, \qquad \mathbf{E} = [\![\mathbf{B}]\!] - [\![\beta]\!] = \begin{bmatrix} [\![0]\!] & [\![4]\!] \\ [\![0]\!] & [\![2]\!] \end{bmatrix}. \tag{B.5}$$

Equation B.4 yields:

$$\begin{bmatrix} [\![0]\!] & [\![2]\!] \\ [\![0]\!] & [\![5]\!] \end{bmatrix} + \begin{bmatrix} [\![2]\!] & [\![1]\!] \\ [\![5]\!] & [\![6]\!] \end{bmatrix} + \begin{bmatrix} [\![0]\!] & [\![5]\!] \\ [\![0]\!] & [\![3]\!] \end{bmatrix} + \begin{bmatrix} [\![3]\!] & [\![3]\!] \\ [\![2]\!] & [\![0]\!] \end{bmatrix} = \begin{bmatrix} [\![5]\!] & [\![4]\!] \\ [\![0]\!] & [\![0]\!] \end{bmatrix}. \tag{B.6}$$

The result of secure matrix multiplication in Equation B.6 is equal to the open matrix product in Equation B.2.

# Secure Control in the Cloud Using Multiparty Computation

**Andrea Victoria Tram Løvemærke**

Control & Automation - Master's Thesis

June 2019


Department of Electronic Systems
Fredrik Bajers Vej 7C
9220 Aalborg Øst