Privacy-preserving Model Predictive Control

Using Homomorphic Encryption

Jurriaan Govers

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Privacy-preserving Model Predictive Control

Using Homomorphic Encryption

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Jurriaan Govers

May 25, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

This thesis is focused on protecting sensitive data in optimization-based control methods. We propose a novel Privacy-preserving Model Predictive Controller (PMPC) where multiple agents are controlled by an untrusted external coordinator. By using the Paillier additively Homomorphic Encryption (HE) scheme, our PMPC allows the coordinator to solve a Quadratic Programming (QP) problem over encrypted data. The PMPC is based on a Projected Gradient Scheme (PGS) on the Lagrange-dual, which enables the use of quadratic cost functions with complicated constraints (e.g., constraints on linear combinations of states and inputs). Compared to the state-of-the-art, the proposed controller protects not only the states and inputs of the agents, but also the system models, cost function and constraints.

Optimization problems with quadratic cost functions and linear constraints, form the basis of a wide class of Model Predictive Control (MPC) problems. Examples of applications are smart-grids, large industrial plants and robotics. To test our PMPC, we focus on the application in self-driving vehicles. Motivated by the AUTOTRAC 2020 competition, we formulate a controller for multiple vehicles in a platoon. An external coordinator controls the longitudinal velocities of up to ten vehicles, while complicated constraints on the positions prevent collisions. By using our PMPC, the external coordinator does not gain access to the private data of the vehicles, protecting their privacy. Because of the wide application domain of MPC, we would like to extend this research in the future by testing the PMPC on other applications as well.

Table of Contents

List of Figures

List of Tables

Acknowledgments

Over two years ago, I started the course Robot Motion Planning and Control by Dr. Javier Alonso-Mora. I enjoyed the course so much that I started discussing the possibilities of doing a thesis in this subject with Javier. He later connected me to Dr. Laura Ferranti, where I started this research in the direction of privacy in control. I am really thankful to Javier for connecting me to Laura, because working with her was more than I could have asked for. Laura was very invested in the project and she was always available to help me, even when she was abroad. I learned a lot from her about the theoretical aspects of the project, but also about organizing my research.

Last November I started working on the AUTOTRAC 2020 competition, under the supervision of Dr. Ing. Sergio Grammatico. I would like to thank the whole team and Sergio specifically, for this opportunity. I learned a lot about the software and hardware we used, which was accelerated by the great help I received from the team.

I would also like to thank Stefan, Alex and Dennis for their feedback on the report. They helped me taking a step back, and distinguish the important information from the unnecessary details, which I find challenging sometimes. The cover of this thesis is made by my sister Floor, the numbers are an encrypted representation of the number 129, which consists of the two primes 3 and 43. Thank you Floor for making the beautiful cover.

The last few months, the world has been a very chaotic place. Due to the pandemic, the final stretch of my time in Delft was different than I imagined. We were not able to go to Italy for the AUTOTRAC competition, I was not able to test my work on the ROSbots, and I had no place to work since the university closed down. I would like to thank, from the bottom of my heart, the support I received from my parents Joke and Niek, and my girlfriend Rachel. Joke and Niek gave me a wonderful place to work, away from the chaos. And Rachel supported me selflessly, for which I am very thankful.

Delft, University of Technology — Jurriaan Govers
May 25, 2020

Chapter 1

Introduction

Optimization-based control methods such as Model Predictive Control (MPC), are powerful tools in applications like smart-grids, large industrial plants and self-driving vehicles. They allow for efficient control, while giving the user the ability to constrain states and inputs. Combining MPC with the computational power of cloud-computing services, enables multiple parties to cooperate and perform complex tasks. One application where the advantages of an external coordinator become apparent, is the coordination of multiple self-driving vehicles. For example, vehicles can coordinate their trajectories to collectively avoid collisions [1, 2, 3], or to create formations [4, 5, 6, 7, 8].

However, using an external party as a coordinator comes at a risk. In order to cooperate efficiently, data must be shared that often contain private information. In the case of vehicles, these data consist of locations, destinations, behavior in traffic or the type of vehicle that is used. To address these issues, control schemes need to be developed that allow cooperation while protecting private data. Therefore, this thesis sets out to answer two questions: First, is it possible to use optimization-based control methods while protecting the privacy of those involved? And second, can an external coordinator use privacy preserving control methods to coordinate the trajectories of multiple self-driving vehicles?

A promising tool for supplying an untrusted controller with private data is Homomorphic Encryption (HE). HE is a class of cryptosystems that allows calculations to be performed over encrypted data, without the need to decrypt them. Literature shows successful applications of HE to control problems like static linear controllers [9, 10, 11, 12]. However, attempts at using HE for optimization-based methods are limited.

Schulze Darup et al. [13] propose a privacy-preserving implementation of explicit MPC for linear systems with linear state and input constraints. In explicit MPC the optimization problem is solved offline, which becomes intractable as the complexity of the controller increases. Solving the optimization problem online (i.e., every time the controller is evaluated) is possible in [14, 15]. Here, a Projected Gradient Scheme (PGS) allows an external coordinator to evaluate an optimization problem. However, it can only evaluate a single optimization iteration per time-step, which is not robust. Furthermore, the constraints in their optimization

problem are limited to box-constraints on the inputs. Schulze Darup et al. [16] extend this to box-constraints on the inputs and states, with the use of the Alternating Direction Method of Multipliers (ADMM). However, the use of complicated constraints (e.g., constraints on linear combinations of states and inputs) is desirable, because it allows to define constraints on the output of the system or constraints coupling the states of agents. Shoukry et al. [17] present another privacy-preserving method for solving optimization problems. It is also based on a PGS but it uses the Lagrange-dual, which allows for multiple optimization-iterations and linear constraints. This makes it a promising candidate to use in MPC. Table 1-1 compares the described methods.

Table 1-1: A comparison of methods for encrypted MPC, as presented in the current literature.

	Online-optimization	Multiple iterations	Complicating constraints	Protects system models
Explicit MPC [13]	-	-	x	-
PGS [14, 15]	x	-	-	-
PGS + ADMM [16]	x	-	-	-
PGS on dual [17]	x	x	x	-

All methods described in the previous paragraph only consider the states and the inputs of agents as private variables. The cost function and constraints of the optimization problem and the system models are considered public. In the example of self-driving vehicles, these data contain private information, such as destinations or the type of vehicle used. In this study we consider as private, any information on: the states and inputs; the system models; the cost function and constraints. We build on the methods for encrypted optimization described in [17], applying them to MPC and extending them to protect these types of sensitive data.

Contribution

In this thesis we investigate the possibility of using optimization-based control methods while protecting the privacy of those involved. The tool we use to protect the privacy of the agents is HE. We consider a scenario where multiple agents use an untrusted external coordinator to solve an optimization problem. The external coordinator should not gain access to any information on: the states and inputs; the system models; the cost function and constraints. The considered models are linear, the cost function quadratic and the constraints are linear combinations of both states and inputs. To test the capabilities of the privacy-preserving controller, a scenario with multiple self-driving vehicles will be considered. The controller should be able to coordinate the trajectories of the vehicles, without gaining access to private information.

Structure of the report

In Chapter 2, the mathematical definitions on which this thesis builds will be explained. It starts with an introduction to Model Predictive Control in Section 2-1. Then it explains the underlying optimization methods in Section 2-2. And finally the Homomorphic Encryption methods are explained in Section 2-3. In Chapter 3, a novel Privacy-preserving Model Predictive Controller (PMPC) is introduced. The application to test this controller on self-driving

vehicles, is formulated in Chapter 4. The results of these tests are presented in Chapter 5, using simulations. Finally, in Chapter 6, conclusions are drawn and recommendations for future work are made.

Chapter 2

Preliminaries

This chapter provides the mathematical framework that forms the basis of this thesis. First, Section 2-1 describes the optimization-based control method Model Predictive Control (MPC). Then Section 2-2 explains the solver that is used to solve the optimization problem provided by the MPC controller. And finally, Section 2-3 explains the cryptographic tools that are used to protect the sensitive data.

2-1 Model Predictive Control

As explained in the introduction, we use the optimization-based control method MPC. Every time this controller is evaluated, an optimization problem is solved to determine the optimal control inputs for the system. The cost function of the optimization problem uses available state measurements, and predicts the behavior of the system for the next N time-steps, where N is called the prediction horizon. The first control input from the resulting control sequence, is applied to the system in closed-loop, and the procedure is repeated in receding-horizon fashion. For more in-depth information about MPC, the reader is referred to [18] and [19].

A linear discrete-time state-space model is considered:

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k), \quad i \in \mathcal{I}_V, \tag{2-1}$$

where $x_i(k) \in \mathbb{R}^n$ represents the states of agent i at time-step $k \geq 0$, $u_i(k) \in \mathbb{R}^m$ the corresponding control inputs, $A_i \in \mathbb{R}^{n \times n}$ and $B_i \in \mathbb{R}^{n \times m}$ the system matrices and $\mathcal{I}_V := \{1, \ldots, V\}$ the set with a total of V agents. To simplify the notation, the assumption is made that all agents have the same number of states ($n_i = n$) and inputs ($m_i = m$).

Prediction matrices

As the name implies, MPC uses a model to predict the evolution of the states. The state evolution $\bar{x}_i$ can be expressed in terms of the current state $x_{0,i} = x_i(0)$ and the input sequence

$\bar{u}_i$, with use of the prediction matrices P_i and S_i, as follows:

$$\underbrace{\begin{bmatrix} x_i(0) \\ x_i(1) \\ \vdots \\ x_i(N) \end{bmatrix}}_{\bar{x}_i} = \underbrace{\begin{bmatrix} I \\ A_i \\ \vdots \\ A_i^N \end{bmatrix}}_{P_i} \underbrace{[x_i(0)]}_{x_{0,i}} + \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ B_i & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_i^{N-1}B_i & A_i^{N-2}B_i & \dots & B_i \end{bmatrix}}_{S_i} \underbrace{\begin{bmatrix} u_i(0) \\ u_i(1) \\ \vdots \\ u_i(N-1) \end{bmatrix}}_{\bar{u}_i}, \tag{2-2}$$

where $\bar{x}_i \in \mathbb{R}^{n(N+1)}$, $P_i \in \mathbb{R}^{n(N+1)\times n}$, $x_{0,i} \in \mathbb{R}^n$, $S_i \in \mathbb{R}^{n(N+1)\times mN}$ and $\bar{u}_i \in \mathbb{R}^{mN}$.

Cost function

To determine the optimal control inputs over the prediction horizon N, an optimization problem is solved. A cost function $\mathcal{V}_i(\bar{x}_i, \bar{u}_i)$, consisting of a stage cost $\ell_i\big(x_i(k), u_i(k)\big)$ and a final cost $\mathcal{V}_i^f\big(x_i(N)\big)$ is defined:

$$\mathcal{V}_i(\bar{x}_i, \bar{u}_i) = \sum_{k=0}^{N-1} \ell_i\big(x_i(k), u_i(k)\big) + \mathcal{V}_i^f\big(x_i(N)\big). \tag{2-3}$$

In this cost function the states and control inputs are penalized with a weighted quadratic cost:

$$\ell_i\big(x_i(k), u_i(k)\big) = \frac{1}{2}\left\|x_i(k)\right\|_{Q_i}^2 + \frac{1}{2}\left\|u_i(k)\right\|_{R_i}^2, \tag{2-4}$$

with the positive semi-definite and symmetric state weight matrix $Q_i \succeq 0 \in \mathbb{R}^{n\times n}$ and the positive definite and symmetric input weight matrix $R_i \succ 0 \in \mathbb{R}^{m\times m}$. The terminal cost is defined as:

$$\mathcal{V}_i^f\big(x_i(N)\big) = \frac{1}{2}\left\|x_i(N)\right\|_{Q_i^f}^2, \tag{2-5}$$

with the positive semi-definite and symmetric final state weight matrix $Q_i^f \succeq 0 \in \mathbb{R}^{n\times n}$. The cost function can now be written as:

$$\mathcal{V}_i(\bar{x}_i, \bar{u}_i) = \frac{1}{2}\bar{x}_i^T \underbrace{\begin{bmatrix} Q_i & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & Q_i & 0 \\ 0 & \dots & 0 & Q_i^f \end{bmatrix}}_{\bar{Q}_i} \bar{x}_i + \frac{1}{2}\bar{u}_i^T \underbrace{\begin{bmatrix} R_i & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & R_i \end{bmatrix}}_{\bar{R}_i} \bar{u}_i, \tag{2-6}$$

with $\bar{Q}_i \succeq 0 \in \mathbb{R}^{n(N+1)\times n(N+1)}$ and $\bar{R}_i \succ 0 \in \mathbb{R}^{mN\times mN}$.

2-1-1 Multi-agent formulation

We consider a scenario where a total of V agents agree on a single cost function. This centralized cost function is minimized by an external coordinator.

Prediction matrices

The state evolutions of the agents are stacked in $\bar{\boldsymbol{x}}$ and can be expressed in terms of the stacked current states $\boldsymbol{x_0}$ and the stacked control inputs $\bar{\boldsymbol{u}}$, in the following way:

$$\underbrace{\begin{bmatrix}\bar{x}_1\\ \bar{x}_2\\ \vdots\\ \bar{x}_V\end{bmatrix}}_{\bar{\boldsymbol{x}}} = \underbrace{\begin{bmatrix}P_1 & 0 & \dots & 0\\ 0 & P_2 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & P_V\end{bmatrix}}_{\boldsymbol{P}}\underbrace{\begin{bmatrix}x_{0,1}\\ x_{0,2}\\ \vdots\\ x_{0,V}\end{bmatrix}}_{\boldsymbol{x_0}} + \underbrace{\begin{bmatrix}S_1 & 0 & \dots & 0\\ 0 & S_2 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & S_V\end{bmatrix}}_{\boldsymbol{S}}\underbrace{\begin{bmatrix}\bar{u}_1\\ \bar{u}_2\\ \vdots\\ \bar{u}_V\end{bmatrix}}_{\bar{\boldsymbol{u}}}, \tag{2-7}$$

where $\bar{\boldsymbol{x}} \in \mathbb{R}^{n(N+1)V}$, $\boldsymbol{P} \in \mathbb{R}^{n(N+1)V \times nV}$, $\boldsymbol{x_0} \in \mathbb{R}^{nV}$, $\boldsymbol{S} \in \mathbb{R}^{n(N+1)V \times mNV}$ and $\bar{\boldsymbol{u}} \in \mathbb{R}^{mNV}$.

Cost function

We define the cost function of the multi-agent system, as the sum of the individual cost functions of the agents:

$$\begin{aligned}\mathcal{V}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}) &= \sum_{i=1}^{V} \mathcal{V}_i(\bar{x}_i, \bar{u}_i) \\ &= \frac{1}{2}\bar{\boldsymbol{x}}^T \underbrace{\begin{bmatrix}\bar{Q}_1 & 0 & \dots & 0\\ 0 & \bar{Q}_2 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & \bar{Q}_V\end{bmatrix}}_{\bar{\boldsymbol{Q}}} \bar{\boldsymbol{x}} + \frac{1}{2}\bar{\boldsymbol{u}}^T \underbrace{\begin{bmatrix}\bar{R}_1 & 0 & \dots & 0\\ 0 & \bar{R}_2 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & \bar{R}_V\end{bmatrix}}_{\bar{\boldsymbol{R}}} \bar{\boldsymbol{u}},\end{aligned} \tag{2-8}$$

with $\bar{\boldsymbol{Q}} \succeq 0 \in \mathbb{R}^{n(N+1)V \times n(N+1)V}$ and $\bar{\boldsymbol{R}} \succ 0 \in \mathbb{R}^{mNV \times mNV}$.

Constraints

An important feature of MPC is the possibility to constrain states and inputs. Using linear combinations of states and inputs, allows to formulate complicated constraints such as output or coupling constraints. An example of a coupling constraint is a collision avoidance constraint on the position of self-driving vehicles. In general, the constraints are defined as:

$$\begin{aligned}\boldsymbol{E_x}\bar{\boldsymbol{x}} &\leq \boldsymbol{e_x},\\ \boldsymbol{E_u}\bar{\boldsymbol{u}} &\leq \boldsymbol{e_u},\end{aligned} \tag{2-9}$$

with $\boldsymbol{E_x} \in \mathbb{R}^{c_x \times n(N+1)V}$, $\boldsymbol{e_x} \in \mathbb{R}^{c_x}$, $\boldsymbol{E_u} \in \mathbb{R}^{c_u \times mNV}$ and $\boldsymbol{e_u} \in \mathbb{R}^{c_u}$, where c_x denotes the number of state constraints and c_u the number of input constraints. In Section 4-2-3, the structure of these matrices is defined for the application of self-driving vehicles.

Optimization problem

The cost function $\mathcal{V}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$ together with the constraints form the following optimization problem:

$$\begin{aligned} \min_{\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}} \quad & \frac{1}{2}\bar{\boldsymbol{x}}^T \bar{\boldsymbol{Q}} \bar{\boldsymbol{x}} + \frac{1}{2}\bar{\boldsymbol{u}}^T \bar{\boldsymbol{R}} \bar{\boldsymbol{u}} \\ \text{s.t.} \quad & \bar{\boldsymbol{x}} = \boldsymbol{P}\boldsymbol{x_0} + \boldsymbol{S}\bar{\boldsymbol{u}} \\ & \boldsymbol{E_x}\bar{\boldsymbol{x}} \leq \boldsymbol{e_x} \\ & \boldsymbol{E_u}\bar{\boldsymbol{u}} \leq \boldsymbol{e_u} \end{aligned} \tag{2-10}$$

2-1-2 Condensed formulation

The equality constraint of Eq. (2-7) allows us to eliminate the states $\bar{\boldsymbol{x}}$ as optimization variables. The cost function is rewritten, such that it depends only on the inputs $\bar{\boldsymbol{u}}$:

$$\begin{aligned} \mathcal{V}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}) &= \frac{1}{2}\bar{\boldsymbol{x}}^T \bar{\boldsymbol{Q}} \bar{\boldsymbol{x}} + \frac{1}{2}\bar{\boldsymbol{u}}^T \bar{\boldsymbol{R}} \bar{\boldsymbol{u}}, \\ &= \frac{1}{2}(\boldsymbol{P}\boldsymbol{x_0} + \boldsymbol{S}\bar{\boldsymbol{u}})^T \bar{\boldsymbol{Q}} (\boldsymbol{P}\boldsymbol{x_0} + \boldsymbol{S}\bar{\boldsymbol{u}}) + \frac{1}{2}\bar{\boldsymbol{u}}^T \bar{\boldsymbol{R}} \bar{\boldsymbol{u}}, \end{aligned} \tag{2-11}$$

$$\mathcal{V}(\bar{\boldsymbol{u}}) = \frac{1}{2}\bar{\boldsymbol{u}}^T \underbrace{\left(\boldsymbol{S}^T \bar{\boldsymbol{Q}} \boldsymbol{S} + \bar{\boldsymbol{R}}\right)}_{\boldsymbol{H}} \bar{\boldsymbol{u}} + \underbrace{\boldsymbol{x_0}^T \left(\boldsymbol{S}^T \bar{\boldsymbol{Q}} \boldsymbol{P}\right)}_{(\boldsymbol{F}\boldsymbol{x_0})^T} \bar{\boldsymbol{u}} + \underbrace{\frac{1}{2}\boldsymbol{x_0}^T \left(\boldsymbol{P}^T \bar{\boldsymbol{Q}} \boldsymbol{P}\right) \boldsymbol{x_0}}_{c}, \tag{2-12}$$

with $\boldsymbol{H} \succeq 0 \in \mathbb{R}^{mNV \times mNV}$ and $\boldsymbol{F} \in \mathbb{R}^{mNV \times nV}$. The scalar $c \in \mathbb{R}$ does not depend on the inputs $\bar{\boldsymbol{u}}$, and is therefore not considered in the optimization problem as defined in Eq. (2-15).

Similarly, the constraints can be expressed in the inputs $\bar{\boldsymbol{u}}$ as well:

$$\begin{aligned} \boldsymbol{E_x}(\boldsymbol{P}\boldsymbol{x_0} + \boldsymbol{S}\bar{\boldsymbol{u}}) &\leq \boldsymbol{e_x}, \\ \boldsymbol{E_x}\boldsymbol{S}\bar{\boldsymbol{u}} &\leq \boldsymbol{e_x} - \boldsymbol{E_x}\boldsymbol{P}\boldsymbol{x_0}, \end{aligned} \tag{2-13}$$

resulting in the following input constraints:

$$\underbrace{\begin{bmatrix} \boldsymbol{E_x}\boldsymbol{S} \\ \boldsymbol{E_u} \end{bmatrix}}_{\boldsymbol{E}} \bar{\boldsymbol{u}} \leq \underbrace{\begin{bmatrix} \boldsymbol{e_x} - \boldsymbol{E_x}\boldsymbol{P}\boldsymbol{x_0} \\ \boldsymbol{e_u} \end{bmatrix}}_{\boldsymbol{e}}, \tag{2-14}$$

with $\boldsymbol{E} \in \mathbb{R}^{(c_x + c_u) \times mNV}$ and $\boldsymbol{e} \in \mathbb{R}^{(c_x + c_u)}$.

Optimization problem

The cost function $\mathcal{V}(\bar{\boldsymbol{u}})$ as defined in Eq. (2-12) is a quadratic in $\bar{\boldsymbol{u}}$ and convex ($\boldsymbol{H} \succeq 0$). The constraints as defined in Eq. (2-14) are linear, making the resulting optimization problem a Quadratic Programming (QP) problem:

$$\begin{aligned} \min_{\bar{\boldsymbol{u}} \in \mathbb{R}^{mNV}} \quad & \frac{1}{2}\bar{\boldsymbol{u}}^T \boldsymbol{H} \bar{\boldsymbol{u}} + (\boldsymbol{F}\boldsymbol{x_0})^T \bar{\boldsymbol{u}} \\ \text{s.t.} \quad & \boldsymbol{E}\bar{\boldsymbol{u}} \leq \boldsymbol{e} \end{aligned} \tag{2-15}$$

2-2 Projected Gradient Scheme on the Lagrange-dual

The core of our MPC formulation is the QP problem of Eq. (2-15). Although a vast amount of methods exist to solve QP problems, most of them are unsuitable due to the limitations caused by the encryption techniques that will be described later.

This thesis builds on the methods presented in [17], where dual optimization is used. Instead of minimizing the primal (i.e., original) problem, dual optimization maximizes the Lagrange-dual. The Lagrange-dual is a function, that gives a lower bound on the optimal value of the primal problem. The maximum lower bound is, under certain conditions, identical to the minimum of the primal problem. For more in-depth information about dual optimization, the reader is referred to [20].

2-2-1 Dual problem

The original optimization problem as defined in Eq. (2-15), is called the primal problem. Maximizing the Lagrange-dual function of the primal problem, is called the dual problem.

Lagrangian

In a Lagrangian, the cost function is augmented by a weighted sum of the constraints. The positive weights μ, are referred to as dual variables. The Lagrangian of Eq. (2-15) is defined as:

$$L(\bar{\boldsymbol{u}}, \mu) = \frac{1}{2}\bar{\boldsymbol{u}}^T \boldsymbol{H}\bar{\boldsymbol{u}} + (\boldsymbol{F}\boldsymbol{x_0})^T\bar{\boldsymbol{u}} + \mu^T(\boldsymbol{E}\bar{\boldsymbol{u}} - \boldsymbol{e}), \tag{2-16}$$

where the number of dual variables $\mu \in \mathbb{R}_{\geq 0}^{(c_x+c_u)}$ equals the number of constraints on the states (c_x) and inputs (c_u). The inputs $\bar{\boldsymbol{u}}$ are called the primal variables.

Lagrange-dual function

In general, the Lagrange-dual function is defined as the infimum of the Lagrangian over the primal variables. The Lagrangian of Eq. (2-16) is quadratic in the primal variables $\bar{\boldsymbol{u}}$, and convex. The infimum is found by taking the gradient with respect to $\bar{\boldsymbol{u}}$, and setting it to zero:

$$\begin{aligned} \nabla_{\bar{\boldsymbol{u}}} L(\bar{\boldsymbol{u}}, \mu) &= \boldsymbol{H}\bar{\boldsymbol{u}} + (\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x_0}), \\ &= 0, \end{aligned} \tag{2-17}$$

resulting in an expression for the optimal $\bar{\boldsymbol{u}}^*$:

$$\bar{\boldsymbol{u}}^* = -\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x_0}). \tag{2-18}$$

When the optimal $\bar{\boldsymbol{u}}^*$ is plugged back into the Lagrangian, the Lagrange-dual function is defined as:

$$\begin{aligned} g(\mu) &= \inf_{\bar{\boldsymbol{u}}} \left(L(\bar{\boldsymbol{u}}, \mu)\right) = L(\bar{\boldsymbol{u}}^*, \mu), \\ &= -\frac{1}{2}(\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x_0})^T\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x_0}) - \mu^T\boldsymbol{e}. \end{aligned} \tag{2-19}$$

Maximization of the dual

The dual problem is defined as the maximization of the Lagrange-dual function $g(\mu)$, over the dual variables μ. The dual problem belonging to Eq. (2-15), is defined as:

$$\max_{\mu \in \mathbb{R}_{\geq 0}^{(c_x+c_u)}} \quad g(\mu) = -\frac{1}{2}(\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x_0})^T \boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x_0}) - \mu^T \boldsymbol{e}. \tag{2-20}$$

2-2-2 Projected Gradient Scheme

Algorithm 1 Projected Gradient Scheme for dual optimization

Preparation

1: $\eta = 1/\sigma_{\max}(-\boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{E}^T)$
2: $\mu^+ = \mu_0$

Optimization

3: **repeat**
4: $\quad \mu = \mu^+$
5: $\quad \nabla g(\mu) = -\boldsymbol{E}\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x_0}) - \boldsymbol{e}$
6: $\quad \mu^+ = \max\{0, \mu + \eta\nabla g(\mu)\}$
7: **until** $\|\mu^+ - \mu\|_\infty \leq \varepsilon$
8: $\mu^* = \mu^+$

Implementation

9: $\bar{\boldsymbol{u}}^* = -\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu^* + \boldsymbol{F}\boldsymbol{x_0})$

The dual problem of Eq. (2-20) is a quadratic concave maximization problem. We propose the Projected Gradient Scheme (PGS) of Algorithm 1, to solve the dual problem. The dual variables μ are iteratively updated in the direction of the gradient, and projected on the feasible set in steps 3 to 7 of the algorithm. The gradient of the Lagrange-dual function of Eq. (2-19) is defined as:

$$\nabla g(\mu) = -\boldsymbol{E}\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x_0}) - \boldsymbol{e}, \tag{2-21}$$

with $\nabla g(\mu) \in \mathbb{R}^{(c_x+c_u)}$. The updated dual variables μ^+ are calculated as:

$$\mu^+ = \max\{0, \mu + \eta\nabla g(\mu)\}, \tag{2-22}$$

where $\max\{.,.\}$ takes the element-wise maximum. We chose the step-size $\eta \in \mathbb{R}_{\geq 0}$ as one over Lipschitz constant [20, Chapter 9.3]:

$$\eta = \frac{1}{\sigma_{\max}(\nabla^2 g(\mu))}, \tag{2-23}$$

where $\sigma_{\max}(.)$ is the maximum singular value. $\nabla^2 g(\mu) \in \mathbb{R}^{(c_x+c_u)\times(c_x+c_u)}$ is the Hessian of the Lagrange-dual function:

$$\nabla^2 g(\mu) = -\boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{E}^T. \tag{2-24}$$

Initial guess

The initial guess to start an optimization problem, influences the number of iterations until convergence is reached. In the case of MPC, the properties of the receding horizon can be used to make a good initial guess. One of those properties is that the problem does not change very much between time steps. One option for the initial guess is therefore to take the result of the previous time-step: $\mu_0(k+1) = \mu^*(k)$.

Another option would be to shift the vector of control inputs $\bar{\boldsymbol{u}}$ by one time-step, and use the predicted values of the state $x(1)$ as $\bar{\boldsymbol{x}}_{\boldsymbol{0}}$. The effectiveness of both methods depend on the type of control problem and need to be investigated per application.

Stopping criterion

The dual problem is a concave maximization problem. It has converged if, for all directions, either the gradient or the corresponding updated dual variable is zero. If this is the case, the updated dual variables μ^+ will identical to the previous ones μ. To verify this condition, we evaluate the infinity norm (i.e., the largest absolute value) of the increment in step 7 of Algorithm 1:

$$\|\mu^+ - \mu\|_\infty \leq \varepsilon, \tag{2-25}$$

where $\varepsilon \in \mathbb{R}_{\geq 0}$ is a tunable threshold.

To tune the threshold ε, a physical interpretation is needed. The dual variables μ^* that are the result of the optimization scheme, are used to find the control inputs $\bar{\boldsymbol{u}}^*$. Using a value other than the theoretical optimal value for μ^* can result in two things. Either the corresponding values of $\bar{\boldsymbol{u}}^*$ are sub-optimal (i.e., they do not minimize the primal cost), or they violate the constraints. Using the the definitions of the gradient (Eq. (2-21)) and the optimal control inputs (Eq. (2-18)), we rewrite the increment of the dual variables:

$$\begin{aligned} \mu^+ - \mu &= (\mu + \eta \nabla g(\mu)) - \mu \\ &= \eta \nabla g(\mu) \\ &= \eta(-\boldsymbol{E}\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x}_{\boldsymbol{0}}) - \boldsymbol{e}) \\ &= \eta(\boldsymbol{E}\bar{\boldsymbol{u}} - \boldsymbol{e}) \end{aligned} \tag{2-26}$$

The last line of this equation gives us a relation between the increment of the dual variables and the constraints. We define a maximum allowed constraint violation:

$$\|\boldsymbol{E}\bar{\boldsymbol{u}} - \boldsymbol{e}\|_\infty \leq \delta, \tag{2-27}$$

which we use to redefine the stopping criterion as:

$$\|\mu^+ - \mu\|_\infty \leq \varepsilon = \eta\delta. \tag{2-28}$$

Sub-optimality can lead to instability of the MPC controller and there are methods to solve this, e.g. by tightening the constraints [21]. However, due to the use of encrypted variables, this approach is not be possible and the allowed sub-optimality δ needs to be tuned by hand.

When the stopping criterion is reached, the updated dual variables μ^+ are returned as the optimal dual variables μ^* in step 8 of Algorithm 1. They define the value of the optimal control inputs $\bar{\boldsymbol{u}}$ in step 9, by using Eq. (2-18):

$$\bar{\boldsymbol{u}}^* = -\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu^* + \boldsymbol{F}\boldsymbol{x_0}). \tag{2-29}$$

2-3 Homomorphic encryption

The technique that is used in this thesis to enable optimization-based control methods with private data is Homomorphic Encryption (HE). HE is a class of encryption techniques that allows certain operations to be done over encrypted data, without the need to decrypt them.

The encryption of a variable m is denoted by $[\![m]\!]$, and its decryption by $\mathcal{D}([\![m]\!]) = m$. We refer to encrypted variables as ciphertext, and to variables that are not encrypted as plaintext.

In *additively homomorphic encryption*, there exists an operation $\oplus$ over encrypted data that represents addition in plaintext: $\mathcal{D}([\![m_1]\!] \oplus [\![m_2]\!]) = m_1 + m_2$. Similarly, in *multiplicatively homomorphic encryption*, there exists an operation $\otimes$ over encrypted data that represents multiplication in plaintext: $\mathcal{D}([\![m_1]\!] \otimes [\![m_2]\!]) = m_1 m_2$. Cryptosystems that allow either of these are called Partially Homomorphic Encryption (PHE) methods and cryptosystems that allow both are called Fully Homomorphic Encryption (FHE) methods. Although it would have great advantages to use FHE over PHE, no practical implementations are currently available. The reader is referred to [22] for an overview of the developments in the area of FHE. An example of commonly used *multiplicatively* PHE methods are El Gamal [23] and RSA [24]. In this thesis the *additively* PHE method of Paillier [25] is used, and will be explained in the following sections.

2-3-1 The Paillier cryptosystem

The Paillier cryptosystem is an asymmetric encryption scheme, meaning that anyone with access to the *public key* can encrypt data, but only the one with access to the *private key* can decrypt them.

Definitions

Let $M = pq$ be a hard-to-factor number, with p and q two large prime numbers. $\mathbb{Z}_M$ denotes the group under addition modulo M, which consist of the set of integers $\mathbb{N}_M = \{0, 1, \ldots, M-1\}$. $\mathbb{Z}_M^*$ and $\mathbb{Z}_{M^2}^*$ denote the groups under multiplication modulo M and M^2 respectively. A multiplicative group consists of the set of relative primes to the modulus: $\mathbb{Z}_M^* = \{a \in \{1, \ldots, M-1\} \,|\, \gcd(a, M) = 1\}$, where the function gcd(.) is the greatest common divisor. Euler's totient function for the group $\mathbb{Z}_M^*$ is defined as $\phi(M) = (p-1)(q-1)$. It gives us the order of the group (i.e., $\phi(M) = |\mathbb{Z}_M^*|$), which is the number of elements in its set.

A plaintext message $m \in \mathbb{Z}_M$, together with a random number $r \in \mathbb{Z}_M^*$, is encrypted by:

$$[\![m]\!] = (M+1)^m r^M \mod M^2, \tag{2-30}$$

where $[\![m]\!] \in \mathbb{Z}^*_{M^2}$, and r is randomly picked for every encryption. The message is decrypted by:

$$\mathcal{D}([\![m]\!]) = \frac{\left[[\![m]\!]^{\phi(M)} \mod M^2\right] - 1}{M} \phi(M)^{-1} \mod M, \tag{2-31}$$

where $\phi(M)^{-1}$ denotes the multiplicative inverse of $\phi(M)$ in $\mathbb{Z}^*_M$.

We refer to the number M as the *public key*, and to its prime roots p and q (or equivalently $\phi(M)$) as the *private key*.

Numerical example

The following numerical example is taken from [26, 11.3.2]:

Let $p = 17$, $q = 11$, $M = pq = 187$ and $M^2 = 34969$. Euler's totient is $\phi(M) = 160$, with multiplicative inverse $\phi(M)^{-1} = [160^{-1} \mod 187] = 90$. Consider encrypting the message $m = 175 \in \mathbb{Z}_{187}$, and decrypting the corresponding ciphertext. Choosing the random number $r = 83 \in \mathbb{Z}^*_{187}$, the ciphertext is calculated as:

$$[\![m]\!] = [\![175]\!] = \left[(1+187)^{175} \cdot 83^{187} \mod 34969\right] = 23911 \in \mathbb{Z}^*_{34969}. \tag{2-32}$$

The decryption is calculated as:

$$\begin{aligned} \mathcal{D}([\![m]\!]) = \mathcal{D}(23911) &= \frac{\left[23911^{160} \mod 34969\right] - 1}{187} 90 \mod 187 \\ &= \frac{25620 - 1}{187} 90 \mod 187 \\ &= 137 \cdot 90 \mod 187 = 175 \end{aligned} \tag{2-33}$$

Level of security

The security of the Paillier cryptosystem relies on the fact that it is computationally hard to retrieve the prime roots of $M = pq$. The level of security is measured in the size, or bit-length, of the primes $p, q \in [2^{l-1}, 2^l - 1]$ with the bit-length $l \in \mathbb{N}$. The latest recommendations on the bit-length [27], advise to take a 2048 [bit] key. As a result, ciphertexts are very larger integers in the order of 2^{4096} or 10^{1232}.

For more information about the computational aspects of the algorithm, the reader is referred to [26] and [28].

2-3-2 Homomorphic properties

As mentioned before, the Paillier cryptosystem is an additively homomorphic cryptosystem, it has the following identities:

$$\forall m_1, m_2 \in \mathbb{Z}_n \quad \text{and} \quad k \in \mathbb{N},$$

$$\mathcal{D}\left([\![m_1]\!](M+1)^{m_2} \mod M^2\right) = m_1 + m_2 \mod M, \tag{2-34}$$

$$\mathcal{D}\left([\![m_1]\!][\![m_2]\!] \mod M^2\right) = m_1 + m_2 \mod M, \tag{2-35}$$

$$\mathcal{D}\left([\![m]\!]^k \mod M^2\right) = km \mod M. \tag{2-36}$$

In other words: the cryptosystem allows for plaintext-ciphertext addition (Eq. (2-34)), ciphertext-ciphertext addition (Eq. (2-35)) and plaintext-ciphertext multiplication (Eq. (2-36)). Note that ciphertext-ciphertext multiplication is impossible, and so are other (non-linear) operations.

Matrix calculations

Let $[\![A]\!]$ denote the encryption of matrix A, where every element of the matrix, $a_{ij} \in \mathbb{Z}_M$, is encrypted individually:

$$[\![A]\!] = \begin{bmatrix[\!]} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix[\!]} = \begin{bmatrix} [\![a_{11}]\!] & [\![a_{12}]\!] \\ [\![a_{21}]\!] & [\![a_{22}]\!] \end{bmatrix}. \tag{2-37}$$

Then the homomorphic properties extend to matrices and vectors, and matrix calculations are straightforward. For example the multiplication of encrypted matrix $[\![A]\!]$ with plaintext matrix B (omitting the modulo M^2 for the sake of readability) would be:

$$[\![A]\!]B = \begin{bmatrix[\!]} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix[\!]} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} [\![a_{11}]\!]^{b_{11}}[\![a_{12}]\!]^{b_{21}} & [\![a_{11}]\!]^{b_{12}}[\![a_{12}]\!]^{b_{22}} \\ [\![a_{21}]\!]^{b_{11}}[\![a_{22}]\!]^{b_{21}} & [\![a_{21}]\!]^{b_{12}}[\![a_{22}]\!]^{b_{22}} \end{bmatrix}. \tag{2-38}$$

2-3-3 Quantization

The Paillier cryptosystem can encrypt messages from the additive group $m \in \mathbb{Z}_M$, in other words, positive integers $0 \leq m < M$. However, the control methods we use are defined over real-valued variables. To overcome this, the variables need to be encoded before they can be used in encryption. In the Python implementation of the Paillier cryptosystem that is used for this thesis, a variant of fixed-point arithmetic is implemented to encode the variables for encryption [29]. The numerical errors due to the quantization for encoding, are expected to be negligible and will not be investigated in detail in this thesis. This assumption is based on the large bit-size used for encryption, and will be verified in Chapter 5 by comparing the results of the encrypted controller to a plaintext implementation. For more information on the subject of quantization refinement, the reader is referred to [30].

Chapter 3

Privacy-preserving Model Predictive Controller

The goal of this thesis is to make a controller for a group of agents, that use an untrusted external coordinator to calculate the optimal control inputs using their sensitive data. The tools that are used for this purpose are the optimization-based control method Model Predictive Control (MPC), a Projected Gradient Scheme (PGS) on the Lagrange-dual for solving the optimization problem and the Paillier cryptosystem for additively Homomorphic Encryption (HE). This chapter introduces a novel Privacy-preserving Model Predictive Controller (PMPC).

The trusted group of agents, that have access to the *private data* and the *private key*, will be called Alice and Bob from here on. The untrusted coordinator will be called Chuck.

Private data

The private data that are used in the MPC problem as defined in Eq. (2-15), consist of any information on: the system matrices A_i, $B_i \, \forall i \in \mathcal{I}_V$; the states $x_i(k) \, \forall i \in \mathcal{I}_V, k \geq 0$; the control inputs $u_i(k) \, \forall i \in \mathcal{I}_V, k \geq 0$; or the state and input constraints $\boldsymbol{E_x}$, $\boldsymbol{E_u}$, $\boldsymbol{e_x}$ and $\boldsymbol{e_u}$.

3-1 Algorithm description

Chuck is an external coordinator, and is used to solve the optimization problem of the MPC method. As explained in Section 2-2, this can be done using a PGS on the Lagrange-dual. This method involves solving two equations iteratively, first the gradient of the Lagrange dual problem:

$$\nabla g(\mu) = -\boldsymbol{E}\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu + \boldsymbol{F}\boldsymbol{x_0}) - \boldsymbol{e}, \tag{3-1}$$

and second, the dual variable update:

$$\mu^+ = \max\{0, \mu + \eta\nabla g(\mu)\}. \tag{3-2}$$

Algorithm 2 Privacy-preserving Model Predictive Controller (PMPC)

Agents		**Coordinator**
Offline		
$\eta = 1/\sigma_{max}(H_\mu)$		
$[\![H_\mu]\!]$, $[\![c_{x_0}]\!]$, $[\![c_e]\!] \leftarrow$ Encrypt H_μ, c_{x_0}, c_e		
Preparation		
1: $\boldsymbol{x_0} \leftarrow$ Measure $x(0)$		
2: $\mu^+ = \mu_0$		
3: $[\![c_\mu]\!] = [\![c_{x_0}]\!]\boldsymbol{x_0} + [\![c_e]\!]$		
4: **Send** $[\![H_\mu]\!]$, $[\![c_\mu]\!]$, η	$\Longrightarrow$	**Receive** $[\![H_\mu]\!]$, $[\![c_\mu]\!]$, η
Optimization		
5: **repeat**		
6: $\mu = \mu^+$		
7: **Send** μ	$\Longrightarrow$	**Receive** μ
	8:	$[\![\nabla g(\mu)]\!] = [\![H_\mu]\!]\,\mu + [\![c_\mu]\!]$
	9:	$[\![\tilde{\mu}]\!] = \mu + \eta[\![\nabla g(\mu)]\!]$
10: **Receive** $[\![\tilde{\mu}]\!]$	$\Longleftarrow$	**Send** $[\![\tilde{\mu}]\!]$
11: $\tilde{\mu} \leftarrow$ Decrypt $[\![\tilde{\mu}]\!]$		
12: $\mu^+ = \max\{0, \tilde{\mu}\}$		
13: **until** $\|\mu^+ - \mu\|_\infty \leq \varepsilon$		
Implementation		
14: $\mu^* = \mu^+$		
15: $\bar{\boldsymbol{u}}^* = -\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu^* + \boldsymbol{F}\boldsymbol{x_0})$		

The private data return in these equations as: the current states $\boldsymbol{x_0}$, the cost matrices $\boldsymbol{H}$ and $\boldsymbol{F}$ (containing information about the system matrices), and the constraint matrices $\boldsymbol{E}$ and $\boldsymbol{e}$. The dual variables μ and the step-size η are not considered private. They only contain indirect information about the system, which is meaningless without any additional (protected) data.

Private implementation

To see if and how Chuck can calculate these equation in a privacy preserving way, we first define the matrix $H_\mu \in \mathbb{R}^{(c_x+c_u)\times(c_x+c_u)}$ and the vector $c_\mu \in \mathbb{R}^{(c_x+c_u)}$:

$$H_\mu = -\boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{E}^T, \qquad c_\mu = -\boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{F}\boldsymbol{x_0} - \boldsymbol{e}, \tag{3-3}$$

so that the gradient can now be written as:

$$\nabla g(\mu) = H_\mu\,\mu + c_\mu. \tag{3-4}$$

Owing to the additively homomorphic properties of the Paillier cryptosystem, it is possible to multiply an encrypted matrix by a plaintext vector, and to add encrypted vectors. If Alice

and Bob send encrypted versions of the matrix $[\![H_\mu]\!]$ and the vector $[\![c_\mu]\!]$ to Chuck, he can calculate the gradient in a privacy-preserving way:

$$[\![\nabla g(\mu)]\!] = [\![H_\mu]\!]\,\mu + [\![c_\mu]\!]. \tag{3-5}$$

However, Chuck is not able to update the dual variables using this encrypted version of the gradient. The order of encrypted variables is generally not preserved, and the max function is therefore impossible to execute over encrypted data. As a solution, we split the gradient ascent step and its projection, and introduce the intermediate variables $\tilde{\mu} \in \mathbb{R}^{(c_x+c_u)}$:

$$[\![\,\tilde{\mu}\,]\!] = \mu + \eta\,[\![\nabla g(\mu)]\!]. \tag{3-6}$$

Chuck calculates the encrypted intermediate variables and sends them back to Alice and Bob, who will decrypt them and perform the projection:

$$\mu^+ = \max\{0, \mathcal{D}([\![\tilde{\mu}]\!])\}. \tag{3-7}$$

These equations correspond to steps 7 to 10 in Algorithm 2.

Preparation

Since encrypting matrices is computationally expensive, it is preferable for Alice and Bob to compute the encryption offline. The matrix H_μ as defined in Eq. (3-3) is a static matrix and its encryption can be calculated offline. However, the vector c_μ depends on the measurement of the current states $\boldsymbol{x_0}$, which changes every time-step. To calculate its encryption efficiently, we split the vector in a part $c_{x_0} \in \mathbb{R}^{(c_x+c_u)\times nV}$ depending on the current state and an independent part $c_e \in \mathbb{R}^{(c_x+c_u)}$:

$$\begin{aligned} c_\mu &= -\boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{F}\boldsymbol{x_0} - \begin{bmatrix} \boldsymbol{e_x} - \boldsymbol{E_x}\boldsymbol{P}\boldsymbol{x_0} \\ \boldsymbol{e_u} \end{bmatrix}, \\ &= \underbrace{\left(\begin{bmatrix} \boldsymbol{E_x}\boldsymbol{P} \\ 0 \end{bmatrix} - \boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{F}\right)}_{c_{x_0}} \boldsymbol{x_0} + \underbrace{\left(-\begin{bmatrix} \boldsymbol{e_x} \\ \boldsymbol{e_u} \end{bmatrix}\right)}_{c_e}. \end{aligned} \tag{3-8}$$

Alice and Bob can now encrypt the vectors c_{x_0} and c_e offline. And they compute the vector $[\![c_\mu]\!]$ every time-step, in step 3 of Algorithm 2, by:

$$[\![c_\mu]\!] = [\![c_{x_0}]\!]\boldsymbol{x_0} + [\![c_e]\!], \tag{3-9}$$

which is far more efficient than encrypting the entire vector c_μ.

Evaluation

Each optimization iteration, Chuck sends an encrypted version of the intermediate variable $[\![\tilde{\mu}]\!]$ to Alice and Bob, who will then decrypt it and calculate its projection μ^+. Alice and

Bob compare the updated dual variable μ^+ to the previous version μ to evaluate the stopping criterion, as defined in Eq. (2-25):

$$\|\mu^+ - \mu\|_\infty \leq \varepsilon. \tag{3-10}$$

This equation can either return positive or negative, where positive means that the largest increment is indeed less than or equal to ε. One of three things can happen now:

1. The equation returns positive, implying that the solver has converged sufficiently. Alice and Bob can now calculate their optimal control inputs as:

$$\bar{\boldsymbol{u}}^* = -\boldsymbol{H}^{-1}(\boldsymbol{E}^T\mu^* + \boldsymbol{F}\boldsymbol{x_0}). \tag{3-11}$$

2. The equation returns negative, implying the solver has not yet converged. However, the number of iterations is higher than a maximum allowed number, or the computation time has surpassed the sample time of the controller. Alice and Bob will calculate their (sub-)optimal control input as mentioned before.

3. The equation returns negative, and both the thresholds on the number of iterations and on the computation time are not surpassed (or not even in place). Then Alice and Bob return the suboptimal μ^+ to Chuck, and he continues to perform another optimization iteration.

Chapter 4

Application

In this chapter we present a possible application of the Privacy-preserving Model Predictive Controller (PMPC). The application is motivated by the AUTOTRAC 2020 competition, where we used a controller to coordinate the longitudinal velocity of four robots. The competition is described in Section 4-1, and in Section 4-2 we introduce the Model Predictive Control (MPC) formulation of a longitudinal velocity controller for multiple vehicles.

4-1 AUTOTRAC 2020

The Joint Research Centre (JRC) of the European Commission organized an event called AUTOTRAC 2020, a competition to promote and develop cooperative autonomous vehicles. Teams of students, universities, research centers and the makers' community were asked to participate with multiple small-scale autonomous vehicles. We entered the competition, which is currently postponed due to the coronavirus pandemic, with a team of students from the TU Delft. Under the supervision of Dr. Ing. S. Grammatico, we used ROSbots to develop our solutions to the competition.

4-1-1 Goals

The goal of the competition is to find out if cooperation between autonomous vehicles can improve the safety and the efficiency of (road-)network utilization. To test these goals, the organizers created two challenges: an urban scenario and a highway scenario. Since the highway scenario is the most relevant with regard to the scope of this thesis, it will be explained in detail below. The complete rules of both scenarios, as well as a technical document with our approach and a description of the ROSbot platform, are included in Appendix A.

Highway scenario

In the highway scenario, four vehicles have to drive around a J-shaped track, representing a single-lane highway. Fig. 4-1 shows a picture of the test track that we made for this scenario,

Figure 4-1: The test track for the highway scenario of the AUTOTRAC 2020 competition, with four ROSbots at the starting position.

with four ROSbots at the starting position. The goal is to increase efficiency by forming a platoon with the vehicles, while avoiding collisions. The efficiency is scored by a combination of distance traveled by the platoon and the time-difference between the first and last vehicle. The safety is scored by subtracting points for collisions. After three minutes, the vehicles have to park in a designated area, shown in green in Fig. 4-1.

4-1-2 Platooning

We designed two separate controllers for each vehicle to coordinate their trajectories, an angular and a longitudinal velocity controller. The angular velocity controller makes sure the vehicle stays within the lane, whereas the longitudinal velocity controller tries to minimize the distance between the vehicles while avoiding collisions.

The leader of the platoon drives at a longitudinal velocity v_{lead}, which is lower than the velocity of the other vehicles v_{follow}. The input of the controller is the distance measured by the sensors at the front of the vehicles. To avoid collisions, the vehicles scale their velocities proportionally with the distance to the vehicle in front. This, in combination with the velocity difference, results in a platoon driving with a predefined inter-vehicle distance d_{platoon}, at the velocity of the leader. Fig. 4-2 shows the input-output relation of the controller.

The controller proved to work sufficient, but not very reliable. Often when the leader-vehicle

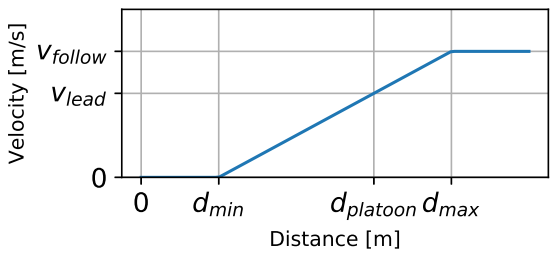


Figure 4-2: Input-output relation of the longitudinal velocity controller, as used in the AUTOTRAC 2020 competition.

decreased its velocity (e.g., to avoid a moving obstacle), collisions occurred at the rear of the platoon. This can be explained by the the fact that the controller only reacts on the vehicle directly in front. This decreases the available reaction-time for the vehicles further back. To ensure the vehicles did not collide, the inter-vehicle distance had to be increased. However, the measurements of all vehicles were available and could be shared by communication. Therefore, the question arose if a centralized multi-agent controller could improve the performance of the platoon. In the following section, we formulate such a centralized multi-agent controller using MPC.

4-2 Longitudinal velocity controller

Motivated by the AUTOTRAC 2020 competition, we design an MPC formulation to coordinate the trajectories of multiple vehicles. An external coordinator controls the longitudinal velocities of the vehicles to form a platoon, while avoiding collisions.

4-2-1 State-space system

The vehicles are modeled by the following discrete-time equations:

$$\begin{aligned} p(k+1) &= p(k) + tv(k), \\ v(k+1) &= v(k) + ta(k), \end{aligned} \tag{4-1}$$

where $p, v, a \in \mathbb{R}$ are the longitudinal position, velocity and acceleration of the vehicle respectively, t the sampling-time and k the current time-step.

The states $x(k) \in \mathbb{R}^2$ are defined as the position and velocity, and the input $u(k) \in \mathbb{R}$ as the acceleration. A state-space representation is defined:

$$x(k+1) = Ax(k) + Bu(k), \tag{4-2}$$

with:

$$\begin{aligned} x(k) &= \begin{bmatrix} p(k) \\ v(k) \end{bmatrix}, & u(k) &= [a(k)], \\ A &= \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}, & B &= \begin{bmatrix} 0 \\ t \end{bmatrix}. \end{aligned} \tag{4-3}$$

We make the assumption that the vehicles are homogeneous, and use this state-space system for all vehicles.

4-2-2 Velocity set-points

Following the control strategy used in the AUTOTRAC 2020 competition, we define two types of vehicles: the leader-vehicle and the follower-vehicles. All vehicles have a static velocity set-

point, v_{lead} or v_{follow} respectively. We translate the states by subtracting these set-points:

$$\tilde{\boldsymbol{x}}_{\mathbf{0}} = \underbrace{\begin{bmatrix} x_{0,1} \\ \vdots \\ x_{0,V} \end{bmatrix}}_{\boldsymbol{x_0}} - \underbrace{\begin{bmatrix} x_{s,1} \\ \vdots \\ x_{s,V} \end{bmatrix}}_{\boldsymbol{x_s}}, \tag{4-4}$$

where $x_{s,1} = [0\ v_{\text{lead}}]^T$ and $x_{s,i} = [0\ v_{\text{follow}}]^T$, $\forall i \in \{2, \dots, V\}$. The translated state $\tilde{\boldsymbol{x}}_{\mathbf{0}}$ is used in the cost function as defined in Eq. (2-12), but not in the constraints.

Since the vehicles are trying to maintain a nonzero velocity, the position of the vehicles will go to infinity as time increases. Therefore, we set the weight on the position to zero:

$$Q_i = \begin{bmatrix} 0 & 0 \\ 0 & q_i \end{bmatrix}. \tag{4-5}$$

To capture the behavior of the tail of the state evolution, the final weight $Q_{f,i}$ is chosen as the solution to the Discrete-time Algebraic Riccati Equation (DARE).

4-2-3 Constraints

We define two types of constraints: coupling constraints and agent specific constraints. Although possible, we define no constraints on the input but only on the states of the vehicles.

Coupling constraints

To prevent collisions, we implement a constraint on the relative positions of the vehicles. The distance between two vehicles i and j, cannot be smaller than a safety distance d_{safe}:

$$p_j(k) \leq (p_i(k) - d_{\text{safe}}). \tag{4-6}$$

The coupling constraints are only defined between neighboring vehicles, and vehicle with the lower index is in front (i.e., $j - i = 1\, \forall i, j > 0$). Fig. 4-3 shows a visual interpretation of this coupling constraint.

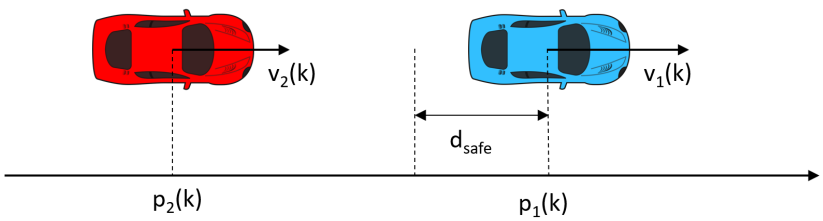


Figure 4-3: Two vehicles with a position $p_i(k)$ and velocity $v_i(k)$. A coupling constraint on the positions ensures that the second vehicle stays a certain safety distance d_{safe} behind the first vehicle.

Eq. (4-6) can be rewritten as a linear state constraint:

$$\underbrace{\begin{bmatrix}-1 & 0\end{bmatrix}}_{E_{c,ij}}\underbrace{\begin{bmatrix}p_i(k)\\ v_i(k)\end{bmatrix}}_{x_i(k)} + \underbrace{\begin{bmatrix}1 & 0\end{bmatrix}}_{E_{c,ji}}\underbrace{\begin{bmatrix}p_j(k)\\ v_j(k)\end{bmatrix}}_{x_j(k)} \leq \underbrace{-d_{\text{safe}}}_{e_{c,ij}} \tag{4-7}$$

By the definition of Eq. (4-1), the input can only influence the positions after two time-steps. The constraints are therefore placed on $\{x_i(2), \dots, x_i(N)\}$ in the following way:

$$\underbrace{\begin{bmatrix}0 & 0 & E_{c,ij} & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \dots & E_{c,ij}\end{bmatrix}}_{\bar{E}_{c,ij}}\underbrace{\begin{bmatrix}x_i(0)\\ x_i(1)\\ x_i(2)\\ \vdots\\ x_i(N)\end{bmatrix}}_{\bar{x}_i} + \underbrace{\begin{bmatrix}0 & 0 & E_{c,ji} & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \dots & E_{c,ji}\end{bmatrix}}_{\bar{E}_{c,ji}}\underbrace{\begin{bmatrix}x_j(0)\\ x_j(1)\\ x_j(2)\\ \vdots\\ x_j(N)\end{bmatrix}}_{\bar{x}_j} \leq \underbrace{\begin{bmatrix}e_{c,ij}\\ e_{c,ij}\\ \vdots\\ e_{c,ij}\end{bmatrix}}_{\bar{e}_{c,ij}} \tag{4-8}$$

with $\bar{E}_{c,ij}, \bar{E}_{c,ji} \in \mathbb{R}^{(N-1)\times 2(N+1)}$ and $\bar{e}_{c,ij} \in \mathbb{R}^{(N-1)}$.

Agent specific constraints

We define the velocity v_{platoon} as the maximum velocity of the platoon, which is lower than the velocity of the follower-vehicles v_{follow}. This ensures that the follower-vehicles are always able to join the platoon. We enforce the maximum platoon velocity at the leader-vehicle as:

$$\underbrace{\begin{bmatrix}0 & 1\end{bmatrix}}_{E_1}\underbrace{\begin{bmatrix}p_1(k)\\ v_1(k)\end{bmatrix}}_{x_1(k)} \leq \underbrace{\begin{bmatrix}v_{\text{platoon}}\end{bmatrix}}_{e_1}. \tag{4-9}$$

The constraint is repeated over the prediction horizon:

$$\underbrace{\begin{bmatrix}0 & E_1 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & E_1\end{bmatrix}}_{\bar{E}_1}\underbrace{\begin{bmatrix}x_1(0)\\ x_1(1)\\ \vdots\\ x_1(N)\end{bmatrix}}_{\bar{x}_1} \leq \underbrace{\begin{bmatrix}e_1\\ \vdots\\ e_1\end{bmatrix}}_{\bar{e}_1}, \tag{4-10}$$

with $\bar{E}_1 \in \mathbb{R}^{N\times 2(N+1)}$ and $\bar{e}_1 \in \mathbb{R}^N$. Note that the constraint is not placed at $x_1(0)$ as this is the measured state, and can not be influenced by the controller.

Combining the agent-specific constraints and the coupling constraints, the matrices $\boldsymbol{E_x}$ and

$\boldsymbol{e_x}$ are defined:

$$\underbrace{\begin{bmatrix} \bar{E}_1 & 0 & 0 & \dots & 0 & 0 \\ \bar{E}_{c,12} & \bar{E}_{c,21} & 0 & \dots & 0 & 0 \\ 0 & \bar{E}_{c,23} & \bar{E}_{c,32} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \bar{E}_{c,(V-1)(V-2)} & 0 \\ 0 & 0 & 0 & \dots & \bar{E}_{c,(V-1)V} & \bar{E}_{c,V(V-1)} \end{bmatrix}}_{\boldsymbol{E_x}} \underbrace{\begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \\ \vdots \\ \bar{x}_{V-1} \\ \bar{x}_V \end{bmatrix}}_{\bar{\boldsymbol{x}}} \leq \underbrace{\begin{bmatrix} \bar{e}_1 \\ \bar{e}_{c,12} \\ \bar{e}_{c,23} \\ \vdots \\ \bar{e}_{c,(V-2)(V-1)} \\ \bar{e}_{c,(V-1)V} \end{bmatrix}}_{\boldsymbol{e_x}}, \tag{4-11}$$

with $\boldsymbol{E_x} \in \mathbb{R}^{N+(N-1)(V-1)\times 2(N+1)V}$ and $\boldsymbol{e_x} \in \mathbb{R}^{N+(N-1)(V-1)}$.

4-2-4 Optimization problem

The resulting optimization problem is:

$$\begin{aligned} \min_{\bar{\boldsymbol{u}}\in\mathbb{R}^{NV}} \quad & \frac{1}{2}\bar{\boldsymbol{u}}^T \boldsymbol{H} \bar{\boldsymbol{u}} + \left(\boldsymbol{F}\tilde{\boldsymbol{x}}_{\boldsymbol{0}}\right)^T \bar{\boldsymbol{u}} \\ \text{s.t.} \quad & \boldsymbol{E}\bar{\boldsymbol{u}} \leq \boldsymbol{e} \end{aligned} \tag{4-12}$$

where $\boldsymbol{E} \in \mathbb{R}^{N+(N-1)(V-1)\times NV}$ and $\boldsymbol{e} \in \mathbb{R}^{N+(N-1)(V-1)}$. The resulting number of dual variables is $n_\mu = N + (N-1)(V-1)$.

Chapter 5

Results

In Chapter 3 we proposed the novel Privacy-preserving Model Predictive Controller (PMPC). To test it, we formulated a longitudinal velocity controller for multiple vehicles forming a platoon in Section 4-2. We simulated this application in Python for an increasing number of vehicles, this chapter shows the results of those simulations. First, the parameters of the controller are explained in Section 5-1. Then the smallest scenario, where only two vehicles are simulated, is analyzed in detail in Section 5-2. The scenario motivated by the AUTOTRAC 2020 competition, where four vehicles were used, is shown in Section 5-3. The scaling of the computation times with respect to the number of vehicles, is investigated in Section 5-4. Finally, we discuss our general observations of the PMPC in Section 5-5.

Platform All simulations were performed on a Windows machine with an Intel Core i7-2630QM 2.00 GHz CPU, and 12 GB of RAM. The code is written in Python, and Pailliers additively Homomorphic Encryption (HE) scheme is implemented using a public library [29].

5-1 Controller parameters

We test the PMPC as a longitudinal velocity controller for multiple vehicles in a platoon. The platoon consists of two types of vehicles: a leader-vehicle and the follower-vehicles. We simulated 300 discrete-time iterations of the controller with a sampling-time of $t = 0.1$ [s], resulting in a total simulated time of $T = 30$ [s].

Set-points The leader-vehicle has a velocity set-point v_{lead} that is lower than the velocity set-point of the follower-vehicles v_{follow}. The values we used in the simulations are:

$$v_{\text{lead}} = 13\,[m/s] \qquad v_{\text{follow}} = 15.725\,[m/s] \tag{5-1}$$

Constraints In Section 4-2-3 we defined two types of constraints: coupling constraints and agent specific constraints. The coupling constraints ensures a safety distance d_{safe} is respected, preventing the vehicles from colliding. The agent specific constraints limit the maximum velocity of the platoon v_{platoon}. The values we used in the simulations are:

$$v_{\text{platoon}} = 14\,[m/s] \qquad d_{\text{safe}} = 10\,[m] \tag{5-2}$$

Weights The cost function of the controller is a weighted quadratic cost on the states and inputs. The weights Q_{lead}, R_{lead} for the leader-vehicle and Q_{follow}, R_{follow} for the follower-vehicles are:

$$\begin{aligned} Q_{\text{lead}} &= \begin{bmatrix} 0 & 0 \\ 0 & 15 \end{bmatrix}, & Q_{\text{follow}} &= \begin{bmatrix} 0 & 0 \\ 0 & 11 \end{bmatrix}, \\ R_{\text{lead}} &= \begin{bmatrix} 1 \end{bmatrix}, & R_{\text{follow}} &= \begin{bmatrix} 1 \end{bmatrix}, \end{aligned} \tag{5-3}$$

where, as explained in Section 4-2-2, the weights on the positions are zero. The corresponding final weights, chosen as the solution to the Discrete-time Algebraic Riccati Equation (DARE), are:

$$Q^f_{\text{lead}} = \begin{bmatrix} 0 & 0 \\ 0 & 46.9 \end{bmatrix}, \qquad Q^f_{\text{follow}} = \begin{bmatrix} 0 & 0 \\ 0 & 39.1 \end{bmatrix}. \tag{5-4}$$

Prediction horizon The prediction horizon is chosen as $N = 10$ which, in combination with the sampling-time of $t = 0.1$ [s], corresponds to a horizon of one second.

Initial states All vehicles start the simulation with a velocity identical to their set-point. The leader vehicle starts at $p_1(0) = 0$ [m]. The second vehicle starts thirteen meters behind the leader $p_1(0) = -13$ [m], and every other vehicle starts another fifteen meters behind the last one $p_i(0) = -(13 + 15(i-1))$ [m] $\forall i \in \{2, \ldots, V\}$.

Stopping criterion In Section 2-2-2, we related the stopping criterion to a maximum allowed constraint violation δ. The chosen value is $\delta = 0.01$, which corresponds to a 0.01 [m/s] violation of the velocity constraints, and a 0.01 [m] violation of the collision avoidance constraints. We consider these violations within safety margins, since we use a safety distance of $d_{\text{safe}} = 10\,[\text{m}]$.

5-2 Two vehicles

The scenario with the minimum number of vehicles $V = 2$, is shown in this section. First, the performance of the controller is shown in Section 5-2-1, after which the computation times of the controller are analyzed in Section 5-2-2.

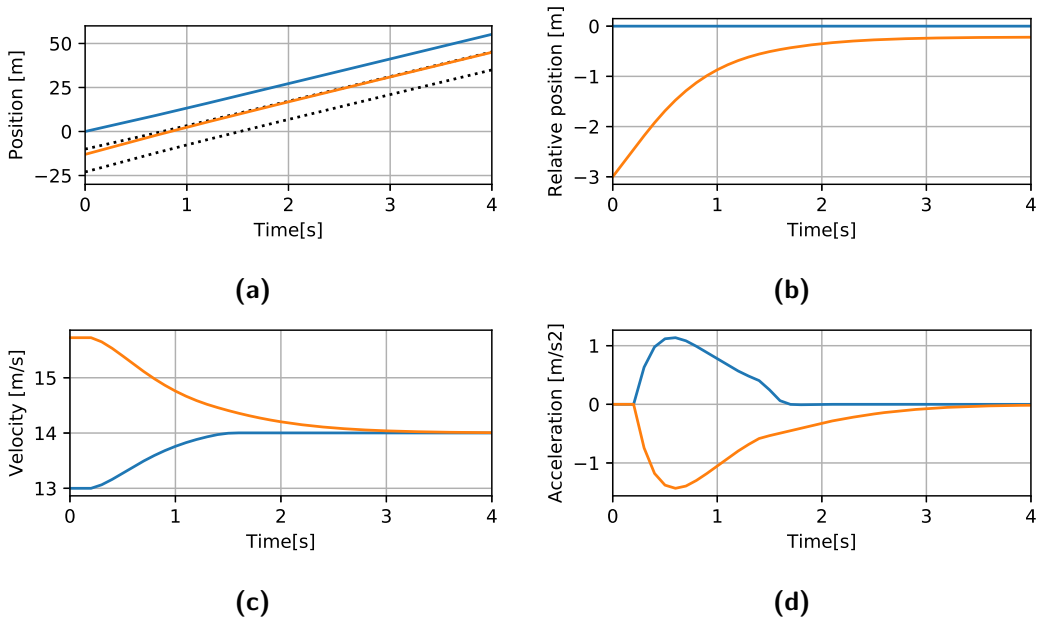


Figure 5-1: Results of a simulation in which the trajectories of two vehicles are coordinated by the PMPC. The leader-vehicle is shown in blue, the follower-vehicle in orange. Fig. (a) shows the positions of both vehicles, the safety distances are shown as black dotted lines. Fig. (b) shows the position of the follower-vehicle, relative to the safety distance of the leader. Fig. (c) shows the velocities of the vehicles, both start at their velocity set-point: the leader-vehicle at 13 [m/s] and the follower-vehicle at 15.725 [m/s]. The velocity of the leader-vehicle is constrained at 14 [m/s]. Fig. (d) shows the acceleration, the control input of the vehicles.

5-2-1 Controller performance

The PMPC should be able to coordinate the trajectories of the two vehicles such that they form a platoon. Fig. 5-1 shows that the velocities of both vehicles successfully converge to the platoon velocity $v_{\text{platoon}} = 14$ [m/s]. Furthermore, the safety distance $d_{\text{safe}} = 10$ [m] is never violated, preventing any collisions. The results of the PMPC were compared to a plaintext implementation of the same controller, to analyze the errors due to quantization. The errors stayed within $|e| < 10^{-13}$, which is negligible as expected. Fig. B-1, in Appendix B, shows the numerical differences between one simulation of the plaintext and the encrypted implementation.

5-2-2 Computation times

Encrypted representations of variables are very large integers, with the current key-size the numbers are in the order of 2^{4096} or 10^{1232}. Performing calculations large of this scale, requires a high computational effort. Computation times are therefore a critical part of the evaluation of any encrypted solver.

Preparation

The preparation that is done by the vehicles, is divided into two parts: the offline preparation, and the preparation that is done every time-step.

Offline The offline preparation consists of generating the encryption key (i.e., finding a suitable pair of large primes $M = pq$) and encrypting the matrices $H_\mu \in \mathbb{R}^{n_\mu \times n_\mu}$, $c_{x_0} \in \mathbb{R}^{2V \times n_\mu}$ and the vector $c_e \in \mathbb{R}^{n_\mu}$. The number of dual variables is defined as $n_\mu = N + (N-1)(V-1) = 19$, resulting in the encryption of $19 \cdot 19 + 4 \cdot 19 + 19 = 456$ variables. In total, the offline preparation phase for this scenario takes on average 9.37 [s].

Online Every time-step, before sending $[\![c_\mu]\!]$ to the coordinator, the vehicles measure their states $\boldsymbol{x_0}$ and calculate: $[\![c_\mu]\!] = [\![c_{x_0}]\!]\boldsymbol{x_0} + [\![c_e]\!]$. This results in $(2V)n_\mu = 4 \cdot 19 = 76$ plaintext-ciphertext multiplications, and the same amount of ciphertext-ciphertext additions. These calculations take on average 0.068 [s].

Comparing the computation times of the two preparation phases, it is apparent that encrypting variables is computationally more expensive than performing matrix calculations. This confirms that splitting c_μ increases the efficiency of the PMPC.

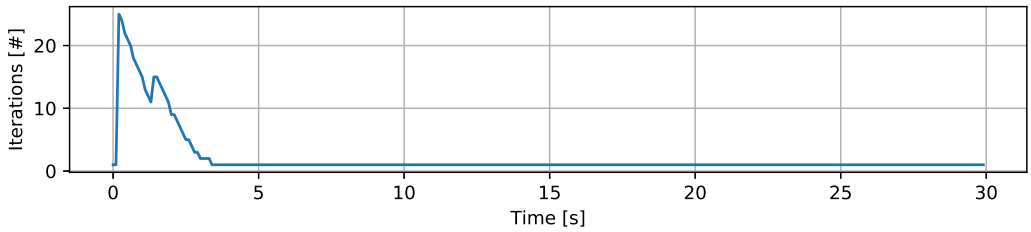
Computation time per time-step

The number of optimization iterations it takes the solver to converge, fluctuates between time-steps. Fig. 5-2a shows a plot of the optimization iterations per time-step. The vehicles are initialized at their velocity set-points with an inter-vehicle distance that is greater than the safety distance. In that case, no constraints are active and the solver only needs a single iteration until it converges. Between 0.1 and 4 seconds, the vehicles approach each-other and more optimization iterations are needed per time-step. After 4 seconds, the velocities converged to v_{platoon} and only a single iteration is needed, since the optimization problem no longer changes between time-steps.

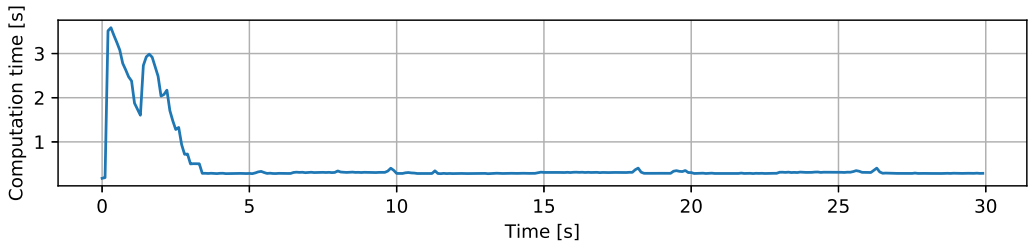
With the number of optimization iterations, the computation time per time-step fluctuates as well. The minimum computation time for a single time-step is 0.1781 [s], corresponding to a time-step where only one optimization iteration is needed. The maximum computation time for a single time-step is 3.5773 [s], which corresponds to the time-step where 25 optimization iterations are needed, the first peak in Fig. 5-2b. As the sampling-time of the controller is $t = 0.1$ [s], this is roughly 36 times slower than real-time. Fig. 5-2c shows the cumulative computation time of the simulation. The simulation takes 144.92 [s] in total, roughly 4.8 times longer than the simulated 30 [s]. Table 5-1 summarizes these numbers.

Table 5-1: The number of optimization iterations and computation time needed for the PMPC to converge in a simulation with two vehicles.

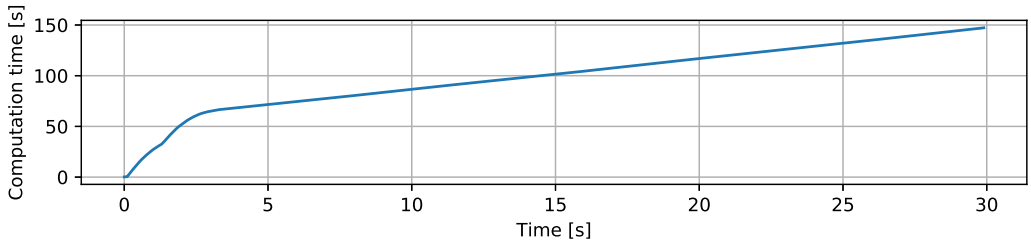
	Optimization iterations [#]	Computation time [s]	Simulated time [s]
Minimum	1	0.1781	0.1
Maximum	25	3.5773	0.1
Total	629	144.9200	30



(a) Number of optimization iterations per time-step



(b) Computation time per time-step



(c) Cumulative computation time

Figure 5-2: Computation time and number of optimization iterations per time-step, for a simulation of two vehicles.

Computation time per optimization iteration

The computation time per optimization iteration can be broken down in two parts. The first part consists of the matrix calculations at the coordinator, and the second part is the decryption of the intermediate dual variables $\tilde{\mu}$ at the vehicles. The two parts together take on average 0.1898 [s].

Coordinator The coordinator calculates the encrypted gradient $[\![\nabla g(\mu)]\!]$ and intermediate dual variables $[\![\tilde{\mu}]\!]$. On average, the computations take 0.0874 [s] per optimization iteration.

Table 5-2: Average computation times per optimization iteration for two vehicles

	Computation time [s]
Coordinator	0.0874
Decryption	0.1024
Total time	0.1898

Vehicles Upon receiving the encrypted intermediate dual variables, the vehicles decrypt them and perform the other calculations over plaintext data. The decryption takes on average 0.1024 [s] per optimization iteration. The calculations over plaintext data (e.g., the projection and evaluating the stopping criterion), take a negligible amount of computation time in comparison to the decryption.

Table 5-2 shows the average computation times per optimization iteration of the different steps. In this scenario the decryption takes slightly longer than the matrix calculations.

5-3 Four vehicles

The design of the longitudinal velocity controller was motivated by the control strategy used in the AUTOTRAC 2020 competition, as explained in Section 4-1. In this section the simulation results of the PMPC will be shown, where we use the same number of vehicles $V = 4$ as in the competition.

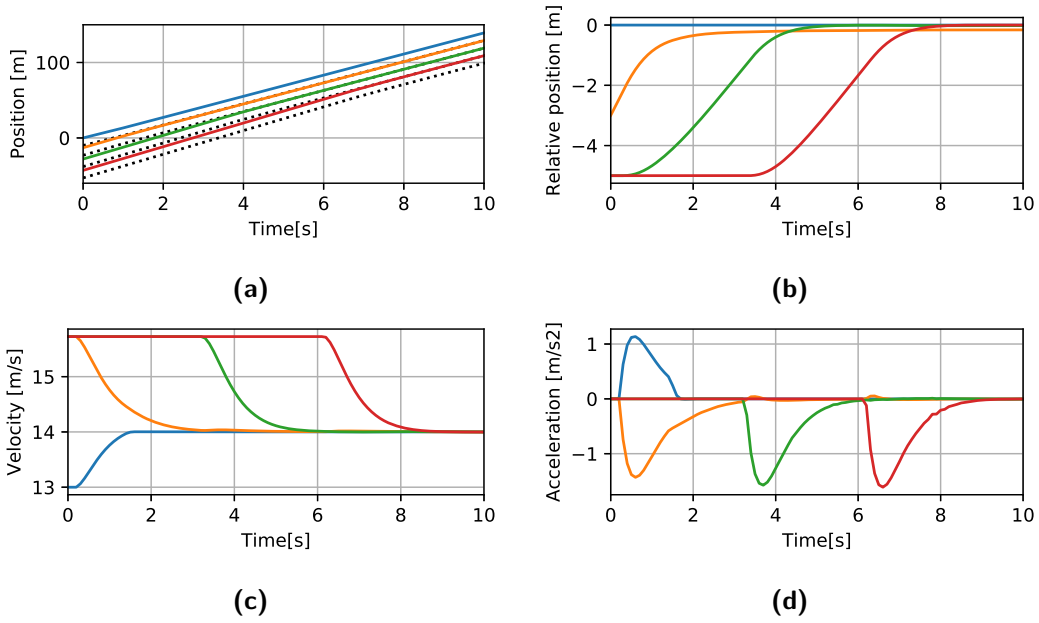


Figure 5-3: Results of a simulation in which the trajectories of four vehicles are coordinated by the PMPC. The leader-vehicle is shown in blue, the first follower-vehicle in orange, the second follower vehicle in green and the third and last follower-vehicle in red. Fig. (a) shows the positions of the vehicles, the safety distances are shown as black dotted lines. Fig. (b) shows the positions of the vehicles, relative to the safety distance of its neighbor. Fig. (c) shows the velocities of the vehicles, the leader-vehicle starts at 13 [m/s] and the follower-vehicle at 15.725 [m/s]. The velocity of the leader-vehicle is constrained at 14 [m/s]. Fig. (d) shows the acceleration, the control input of the vehicles.

5-3-1 Controller performance

Fig. 5-3 shows the trajectories of the leader-vehicle and the three follower-vehicles. Every time a follower-vehicle reaches the platoon, the PMPC successfully reduces its velocity to $v_{\text{platoon}} = 14$ [m/s]. The only time the collision avoidance constraint is violated is when last follower-vehicle joins the platoon. Between the 8.7 and 9.7 second mark, the safety distance of $d_{\text{safe}} = 10$ [m] between vehicles 3 and 4 is exceeded with a maximum of 0.001 [m]. This is well below the allowed constraint violation of $\delta = 0.01$ [m]. Similar to the scenario with two vehicles, the errors due to quantization are within $|e| < 10^{-13}$. Fig. B-2, in Appendix B, shows the differences between the plaintext and the encrypted implementation.

5-3-2 Computation times

The number of vehicles influences the size of the control problem, and with it the computation times. This section will show the computation times for a simulation of four vehicles controlled by the PMPC.

Preparation

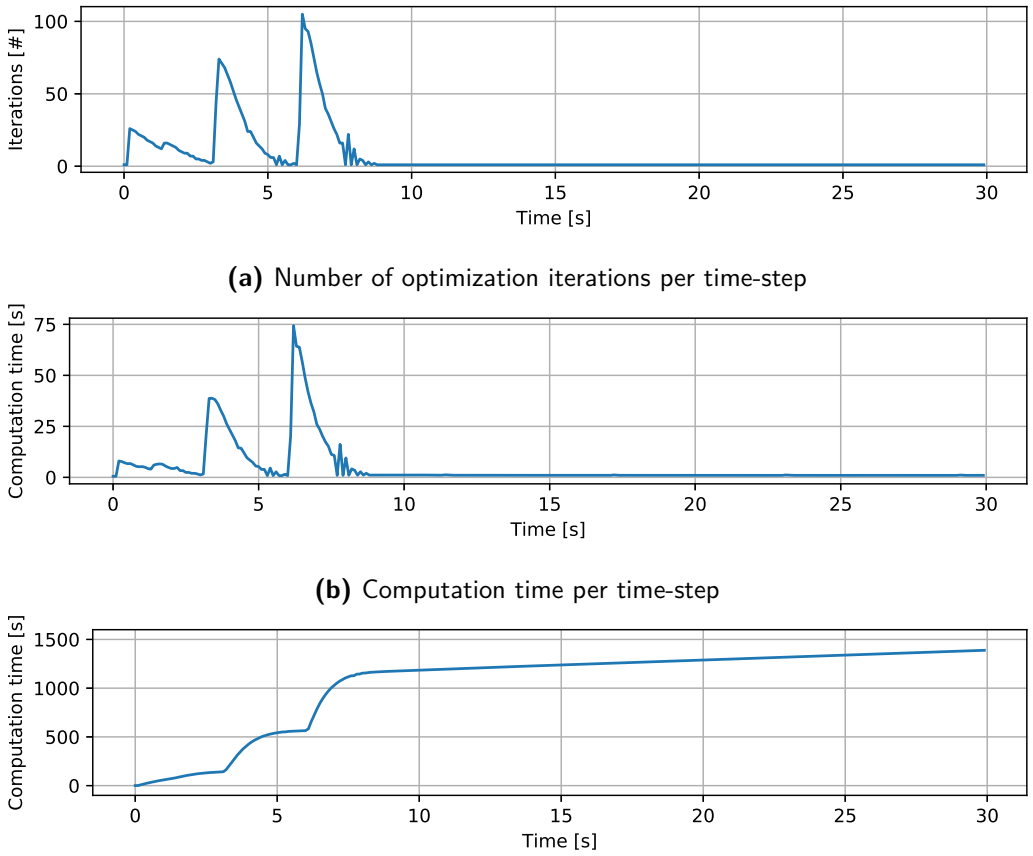
The number of dual variables for this scenario is $n_\mu = N + (N-1)(V-1) = 37$.

Offline The number of variables that are encrypted offline is defined as: $n_\mu^2 + (2V)n_\mu + n_\mu = 37^2 + 8 \cdot 37 + 37 = 1702$. This, in combination with generating the encryption key, takes on average 34.93 [s]. Due to the quadratic dependency on n_μ, this is significantly higher than the 9.37 [s] of the scenario with two vehicles.

Online The number of plaintext-ciphertext multiplications and ciphertext-ciphertext additions to calculate $[\![c_\mu]\!]$ is: $(2V)n_\mu = 8 \cdot 37 = 296$. On averages, this takes 0.2477 [s] every time-step.

Computation time per time-step

Fig. 5-4a shows the number of optimization iterations per time-step. Three peaks are clearly visible, corresponding to the three follower-vehicles joining the platoon. The peaks increase in height as the number of vehicles in the platoon increases. This increase is even more visible in Fig. 5-4b, where the computation time per time-step is shown. At the start of the simulation, only the constraints on the leader-vehicle and the first follower-vehicle are active. As a result, the dual variables corresponding to the rest of the constraints are zero. When a ciphertext variable is multiplied by a plaintext zero, the result is automatically set to zero without any ciphertext calculations. This reduces the computation time significantly. As more vehicles join the platoon, the number of nonzero dual variables increases and with it the computation time.



(a) Number of optimization iterations per time-step

(b) Computation time per time-step

(c) Cumulative computation time

Figure 5-4: Computation time and number of optimization iterations per time-step, for a simulation of four vehicles.

The minimum computation time for a single time-step is 0.5399 [s], corresponding to a time-step where only one iteration is needed. The maximum computation time for a single time-step is 74.4880 [s], corresponding to a time-step where 105 optimization iterations are needed. The maximum computation time is roughly 745 times slower than real-time, and significantly higher than the scenario with two vehicles, where the maximum is 3.5773 [s]. The simulation takes 1392.45 [s] in total, roughly 46 times longer than the simulated 30 [s] and roughly 10 times longer than the simulation with two vehicles. Table 5-3 summarizes these numbers.

Table 5-3: The number of optimization iterations and computation time needed for the PMPC to converge in a simulation with four vehicles.

	Optimization iterations [#]	Computation time [s]	Simulated time [s]
Minimum	1	0.5399	0.1
Maximum	105	74.4880	0.1
Total	2230	1392.4505	30

Computation time per optimization iteration

On average, the computation time per optimization iteration for this scenario is 0.5889 [s].

Coordinator The coordinator takes on average 0.3794 [s] per optimization iteration to perform the matrix calculations over encrypted data.

Vehicles The decryption takes on average 0.2095 [s] per optimization iteration. The rest of the calculations take a negligible amount of computation time.

Table 5-4 shows the average computation time of the different steps per optimization iteration. As opposed to the simulation with two vehicles, the computation time at the coordinator exceeds the computation at the vehicles.

Table 5-4: Average computation times per optimization iteration for four vehicles

	Computation time [s]
Coordinator	0.3794
Decryption	0.2095
Total time	0.5889

5-4 Scaling

To investigate how the computation times of the PMPC scale, we simulated several scenarios, increasing the number of vehicles. The previous sections showed the detailed results of the scenarios with two and four vehicles. In this section we use the results of all simulated scenarios to analyze how the computation times scale.

Preparation

The computation time needed for the two preparation phases, depends on the number of dual variables n_μ. This is defined as $n_\mu = N + (N-1)(V-1) = 9V + 1$, and scales linearly with the number of vehicles.

Offline Although the offline preparation is not time-sensitive, it does take a significant amount of time to encrypt the matrices. The number of variables to be encrypted is $n_\mu^2 + (2V)n_\mu + n_\mu$, which increases quadratically with the number of vehicles. Fig. 5-5 shows the computation times, and the quadratically fitted curve confirms this relation.

Online The number of plaintext-ciphertext multiplications and ciphertext-ciphertext additions to calculate $[\![c_\mu]\!]$ is: $(2V)n_\mu$. The computation time should therefore scale quadratically as well.

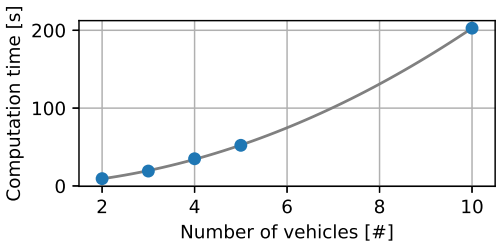


Figure 5-5: Offline preparation

Computation time per optimization iteration

The number of matrix calculations over encrypted data scales faster than the number of decryption operations. Already in the scenario with four vehicles, the computation time needed by the coordinator exceeded the computation time needed for decryption.

Coordinator The number of multiplications and additions the coordinator needs to perform over encrypted data is $n_\mu^2 + n_\mu$, which is quadratic in the number of vehicles. Fig. 5-6a shows the corresponding computation times, and confirms this quadratic trend.

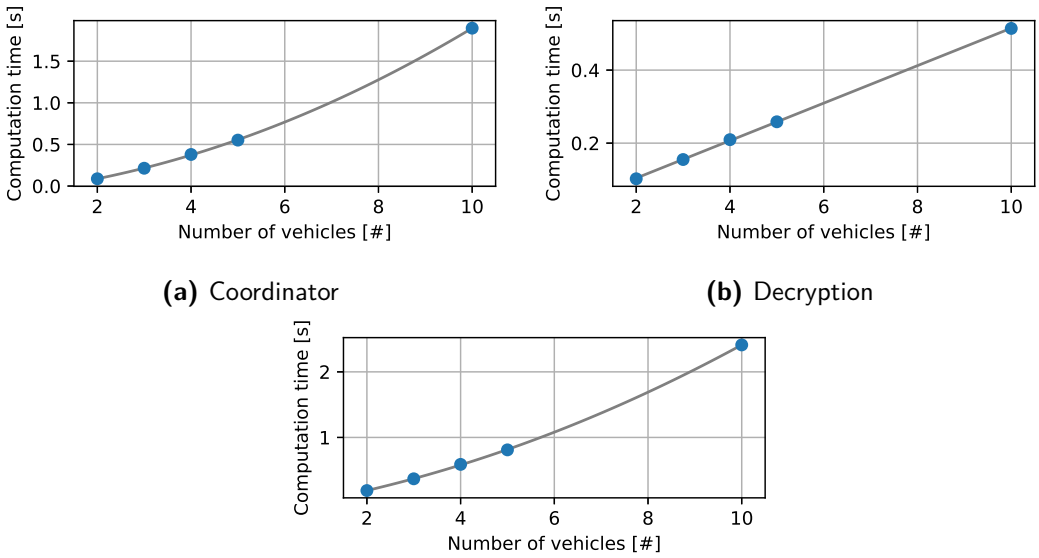
Vehicles The number of variables to be decrypted is the number of dual variables, n_μ, and scales linearly with the number of vehicles. Fig. 5-6b shows the corresponding computation times, and clearly confirms this linear trend.

The total computation time per optimization iteration is the sum of the coordinator and the vehicles. Fig. 5-6c shows the corresponding computation times, and confirms the resulting quadratic trend.

Iterations until convergence and total simulation time

The total number of iterations needed for the PMPC to converge, increases with the number of vehicles as well. Every time a vehicle joins the platoon, a peak in optimization iterations is visible. Fig. 5-7a shows the total number of performed optimization iterations, for an increasing number of vehicles. To formulate a scaling relation is nontrivial. In the case of the first few vehicles, the number seems to scale quadratically. However, as the number of vehicles increases the curve flattens to a line.

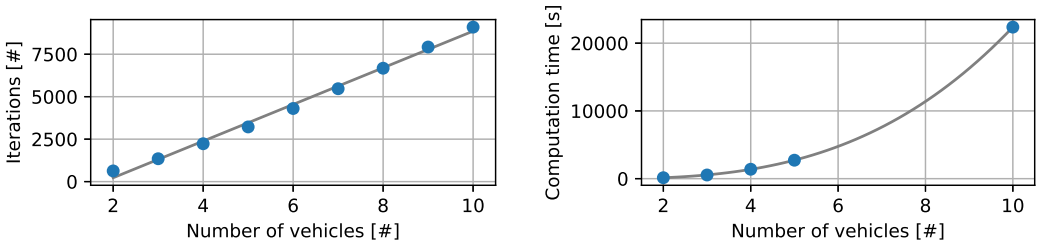
The computation time of the complete simulation is the product of the computation time per optimization iteration and the number of needed optimization iterations. Fig. 5-7 shows the total computation times for the complete simulations, the scaling relation is approximated by a third order polynomial.



(a) Coordinator **(b)** Decryption

(c) Total

Figure 5-6: Computation times per optimization iteration for increasing number of vehicles



(a) Total number of iterations **(b)** Total simulation time

Figure 5-7: Computation times for increasing number of vehicles

5-5 Discussion

This chapter showed that the proposed PMPC is capable of coordinating the longitudinal velocities of up to ten vehicles in an platoon. However, it took more than six hours to complete a simulation with ten vehicles. This is almost 750 times slower than the simulated 30 [s]. The matrix calculations performed by the external coordinator, take up roughly 75% of that time. The simulations were run on a standard, almost ten year old, laptop. However, in practice the coordinator could be a cloud server with specialized GPUs. Combining this with fact that the matrix calculations are highly parallelizable, can improve the computation times. Because we used a single PC for the simulations, the vehicles decrypt all the dual variables in series. However, in practice the vehicles only need to decrypt the dual variables that apply to them, which distributes the computational burden.

As the number of vehicles increases, the control inputs start to become noisy. Fig. C-1, in Appendix C, shows the results of the simulation with ten vehicles, where this is clearly visible. This behavior occurs because the solver uses values of μ^* that are sub-optimal. By decreasing

the value for δ in the stopping criterion, the noisy behavior disappears. However, this comes at the cost of a much higher number of needed optimization iterations, which increases the computation time of the PMPC.

The methods presented by Schulze Darup et al. [15, 16], rely on a Projected Gradient Scheme (PGS) where only a single optimization iteration can be used per time-step. This is not a robust solution, especially when the complexity of the control problem increases. Our PMPC is capable of using this strategy, but Fig. C-2, in Appendix C, shows that that the results of this strategy are undesirable.

Chapter 6

Conclusions and future work

This thesis set out to answer two questions: first, is it possible to use optimization-based control methods while protecting the privacy of those involved? And second, can an external coordinator use privacy preserving control methods to coordinate the trajectories of multiple self-driving vehicles?

This chapter will answer those questions in Section 6-1, and make recommendations for future work in Section 6-2.

6-1 Conclusions

To answer the first question, we proposed a novel Privacy-preserving Model Predictive Controller (PMPC). The PMPC protects sensitive data of multiple agents, by encrypting them before sending them to an external coordinator. We used the additively homomorphic properties of the Paillier cryptosystem, to allow the external coordinator to perform calculations over the encrypted data. By using a Projected Gradient Scheme (PGS) on the Lagrange-dual, multiple optimization iterations of a quadratic cost function can be evaluated. This is an improvement with respect to comparable methods in the literature. Furthermore, the PMPC can constrain linear combinations of inputs and states in the optimization problem. This enables complicated constraints such as output or coupling constraints. By design, the external coordinator does not gain access to any information on: the states and inputs of the agents; the system models; the cost function and constraints.

In the second part of this thesis, we formulated a longitudinal velocity controller for multiple self-driving vehicles in a platoon. This formulation was used to test the capabilities of the PMPC. Using simulations, we showed that the controller successfully coordinated the trajectories of the vehicles to form a platoon. The fact that the PMPC allows linear constraints, enabled the vehicles to avoid collisions. Furthermore, by comparing the encrypted controller to a plaintext implementation, it can be concluded that no concessions to the accuracy of the controller were made. We can also conclude that the calculations over encrypted data are

computationally expensive. Therefore, additional research on this topic is needed, before the PMPC can be used in practice by self-driving vehicles.

In conclusion, the results of this research in privacy-preserving optimization-based control, are very promising. We showed with the PMPC that encrypted controllers can protect more data compared to the state-of-the art, while allowing a wider class of control problems. However, we also recognize that there is still room for improvement, and we will make recommendations for future work in the following section.

6-2 Future work

Incorporating cryptography in the design of controllers is a relatively recent development. There is still a lot to learn and discover in this area, especially since the social relevance of privacy in technology is increasing. During this research we encountered several interesting possibilities, that are listed here as recommendations for future work:

1. *Privacy-preserving comparisons.* The proposed PMPC uses a PGS to solve the underlying optimization problem. The projection-step of this method is currently impossible to perform over encrypted data. However, several studies introduce a system that compares two encrypted values and outputs the greatest of the two [31, 32]. It is worth to investigate if these systems can be used to offload the projection-step to the external coordinator. This allows the coordinator to evaluate multiple optimization iterations, decreasing the computational burden for the agents.

2. *Secure Multi-Party Computation (SMPC).* In the current design of the PMPC, a single encryption key is used to encrypt all sensitive data. Only the external coordinator is considered malicious, while the other agents are trusted. SMPC is an area of cryptography, considering computations with multiple untrusted parties [33]. The combination of privacy-preserving optimization with SMPC techniques such as Secret Sharing [34], or SPDZ [35], is very promising.

3. *Other applications.* The PMPC was tested for motion control of self-driving vehicles. Since Model Predictive Control (MPC) is suitable for a wide variety of control problems, the controller should also be tested for other applications. For example, managing energy prices in smart-grids, where sensitive data consist of energy consumption and generation.

4. *Sequential Quadratic Programming (SQP).* The optimization problems that the PMPC is capable of solving, consist of quadratic cost functions with linear constraints. Although this is a step forward compared to the current literature, many control applications demand nonlinear optimization problems [2, 3]. The possibility of using SQP to approximate nonlinear optimization problems, could be investigated.

5. *Dedicated hardware.* Computation times often form the bottleneck of privacy-preserving controllers. Hardware implementations of Homomorphic Encryption (HE) schemes show significant improvement in terms of computation time, when compared to software implementations [36]. It is worth investigating if, and how, dedicated hardware can improve the PMPC.

6. *Parallelization.* A major advantage of using an external coordinator in general, is the computational power of such services. This computational power is often obtained by distributing computations over several cores or servers. As matrix operations are highly parallelizable, implementing the coordinator calculations of the PMPC in parallel is possible. In the current implementation, the computation time of the coordinator scales quadratically with the number of agents. By distributing the calculations, computation times can be kept constant.

7. *Physical experiments.* The capabilities of the PMPC were tested using simulations in Python. These simulations are simplified representations of reality. Testing on physical systems, such as the ROSbot platform, leads to a better understanding of factors such as: measurement noise, communication delays and model errors. Physical experiments with privacy-preserving controllers are rarely covered in literature, and have a high added value.

Appendix A

AUTOTRAC 2020, additional documents

A-1 ROSbots

As a robotic platform for the AUTOTRAC 2020 competition, we used Husarion ROSbots, a 200x235x220 mm [LxWxH] mobile robot. The ROSbots are equipped with four wheels and DC motors with encoders, an RGB(D) camera, a LIDAR, inertial sensors and four infrared distance sensors (two at the front and two at the back). The basis is a CORE2-ROS controller with an Asus Tinker Board, running a robust and light version of Ubuntu 16.04. The ROSbots are programmed using ROS Kinetic.

Terms and Conditions (Rev.02)
AUTOTRAC 2020 – DG JRC, EC

1. Introduction

Connectivity and automation are expected to bring a deep transformation in the automotive industry and the transportation networks. The European Commission took a prominent role in the deployment of connected driving, by setting up a C-ITS Deployment Platform on 2014 (European Commission, 2014). In the final report of the C-ITS Platform (European Commission, 2017) it is mentioned that for cities across the EU, the potential arrival of automation raises the prospect of safety issues, increased traffic and consequently worsened pollution and congestion, if not tailored and shaped towards the needs of local authorities. A main report conclusion is the need for orchestration of services, that is, to put in place, when appropriate, the traffic management measures by the means of the different stakeholders, public and private, that are required to come together and act accordingly to pre-established agreements. However, the impact is still vague as current Automated Vehicles (AV) products do not focus on vehicles' cooperation (System-centric approach) but rather on individual performances (User-centric approach). Transport engineers agree that better ways of traffic management need to be investigated and JRC conducts already research towards this direction through the C2ART project[1].

The organization of a small-scale AUTOnomous vehicle TRAffic Challenge (AUTOTRAC) is funded by the JRC Exploratory Research Programme (ER activity). It aims to raising awareness on the potential impact of automated vehicles' cooperation in future transport networks. Are CAVs capable to do a better utilization of the network? Can they deliver the anticipated promises for less congestion, safer and greener mobility? **Promotion of the cooperation between AVs, more efficient energy consumption and the reasonable utilization of the transport network are the main pillars of the AUTOTRAC initiative.** AUTOTRAC is a competition of small-scale AVs asking the participation of students, universities, research centers and the makers' community. Each team participates with multiple AV robots that should demonstrate a) efficiency in automated driving and b) communication with the other AVs of the fleet for better network utilization. So, in contrast to currently running small-scale AV competitions, AUTOTRAC promotes the conceptualization and implementation of collaborative behaviors between robot vehicles.

2. General information

The competition is organized by the Joint Research Center (JRC) of the European Commission and it will take place in Ispra (VA) on March 30th to April 1st 2020.

The application is now closed.

Seven teams have confirmed their participation. By confirming their participation, the teams agree with the terms and conditions of the competition.

Each team selected for participation in the JRC AUTOTRAC should provide specific documentation by the 15th of February 2020 (see qualification section).

The technical specifications will freeze one month before the competition. From the moment that the list of participant is fixed, any updates will be communicated via email and not via the website. **For further questions regarding the JRC AUTOTRAC please contact JRC-AUTOTRAC@ec.europa.eu**

[1] https://trimis.ec.europa.eu/project/towards-connected-coordinated-and-automated-road-transport-c2art-system

3. Participation

Admission to the AUTOTRAC competition is subject to the following conditions:

- **Up to 4 members of each team are allowed** to attend the competition in the Ispra premises of the JRC. However, there is no restriction the individuals that contribute in the preparation of the team. Each team should participate with exactly 4 vehicles.
- **The first, second and third team will be awarded with the amount of 3.000€, 2.000€, 1.000€ respectively. Furthermore, for the winning team of the challenge there will be the possibility to develop a demonstrator in the Ispra site of the JRC with a contract of around 15.000€. Finally, the winning team will have the opportunity to demonstrate their performance during the 2020 Transportation Research Arena conference, which will be held on April 27-30, 2020 in Helsinki (https://traconference.eu/). The cost for participating to the conference will be covered by the JRC.**
- **The JRC will provide financial support for travel and accommodation for all the teams. The financial support will be provided according to the rules of the European Commission. More details will be communicated at a later stage.**
- **There is no admission fee for the competition**
- Participants must be over 14 years old at the start of the competition. At least one of the team member shall be at least 18 years old.
- Each team will have to indicate a contact person who will interact with the organizers of the competition.
- There is no possibility of having additional visitors each team unless discussed explicitly with the organizers.
- A technical area will be available to the teams near the competition area with 220V electric power supply.
- By registering, every team and every participant declare their agreement with the publication of image, video and audio recordings. This also includes the recording of team presentations.
- JRC will offer to all participants snacks, coffee and lunch, while the first day there will be a dinner in the JRC premises.
- The participants accept a JRC referee team that will decide the final ranking and will be responsible to solve any objections or problems that may arise.

4. Vehicle requirements

This section describes the regulations related to the vehicle requirements. Violation of these requirements can lead to exclusion of the competition.

For each vehicle the following should apply:

- The vehicle must be registered before the competition. The registration process includes technical inspection of the vehicle, marking the vehicle with a color sticker and testing of the remote start and stop function.
- Technical inspection must be completed by the time specified by the organizers.
- The vehicle must not damage the field or endanger the spectators in any way.
- The vehicle must be completely automated. Any kind of interaction with the team members or any other remote entity during the competition is forbidden.
- The vehicle must have a remote from where the vehicle can be only started or stopped.
- The vehicle should be equipped with electric motors.

- Energy must be supplied with batteries. Max Voltage battery 24V or LiPo 6S (22,2V).
- Changing and charging batteries is allowed.
- The vehicles must be based on a chassis with maximum dimensions 150x200 x200mm (width x length x height).
- The vehicle's maximum weight should be 3kg.
- The sensor setup can be arbitrarily chosen by the teams. Laser sensors are allowed only up to class 2 devices.
- Reference standard for laser devices: IEC 60825-1: 2014
- Only sensors for external lines/spaces recognition can exceed the dimensions of the vehicle by a maximum of 3 cm on all sides.
- There is no limitation to the use of robotic platforms and sensors.
- Each vehicle must be equipped with at least one camera for environment recognition.
- It is possible to use smartphones or tablets (not PCs) beyond the maximum measurements and weight of the rules.
- During the competition (2nd and 3rd day) the hardware of the vehicle must not be modified except in case of supervised repair.
- The software can be changed during the competition (2nd and 3rd day), but not during the run
- To facilitate the recognition of the 4 vehicles by the judges and the camera, a sticker with a different color (red, green, blue, yellow - 3cm diameter) will be provided by the organization. The vehicle must provide a flat space on top to position this sticker.
- Each team is free to apply Vehicle-to-Vehicle (V2V) connectivity between its vehicles. No WiFi network for such reason will be provided by the JRC.

5. Tracks

This section describes the specifications of the track where the scenarios will take place. There will be 2 tracks.

J-shape track: The J-shape track, resembles highway traffic conditions. It is a one-lane track with the shape of the letter 'J'. Regarding its dimensions, the track will be enclosed in a rectangle of 4x3 meters (4 panels 2x1,5 meters joined together with insulating tape). The width of the lane is 300mm.

The minimum turning radius of the line is 400mm (in the middle of the lane).

The track will be crossed by a vehicle (obstacle) that will move forward and back along a white line (like a tram). If the vehicle touches the obstacle, it will be penalized. The size of the mobile obstacle is approximately 150x150x150mm.

On the sides of the path there is the parking area, printed in green ink. In the parking space there will be no line markings. The challenge is to park all vehicles without touching each other and the whole vehicle body being inside the green area.

Each vehicle is eligible to move freely within the road surface. An illustrative representation of the track is depicted in Fig. 1 (left).

#-shape track: The #-shape represents urban traffic conditions. It is a one-lane track with 12 intersections. Regarding its dimensions, the track will be enclosed in a rectangle of 4x3 meters (4 panels 2x1,5 meters joined together with insulating tape). The width of the lane will be 250mm. An illustrative representation of the track is depicted in Fig. 1 (right).

The black lane is surrounded by 35 cm of free-white space on both sides. The minimum space from the outer edges is 25 cm.

On the #-shape track, there will be test areas for the colors and the type of traffic signs.

The material for both tracks will be rigid panels (FOREX thickness 5mm), with white background and black the surface of the road/lane. Everything will be printed in four-colour ink process (CMYK).

The colors used in the tracks and in the road signals are:

White	CMYK: 0, 0, 0 ,0	RGB: 255, 255, 255
Black	CMYK: 0, 0, 0, 100	RGB: 0, 0, 0
Red	CMYK: 0, 100, 100, 0	RGB: 255, 0, 0
Green	CMYK: 100, 0, 100, 0	RGB: 0, 255, 0
Blue	CMYK: 100, 100, 0, 0	RGB: 0, 0, 255
Yellow	CMYK: 0, 0, 100, 0	RGB: 255, 255, 0

6. Scenarios and objectives

There are two test scenarios in the JRC AUTOTRAC competition, one scenario corresponding to highway conditions, another to the urban driving. The layout of the two test tracks is shown in Figure 1.

Highway Scenario and parking (J-shape track)

The Highway scenario takes place on the J-shape track. Each team sets up all four vehicles in the track. The 4 vehicles will be positioned on the red START line in a straight line. Intervehicle distance will be set to 5 cm. At their starting position, the vehicles must be positioned with both wheels on a portion of the black track; only the sensors (extra 3 cm) can be on the white background. The vehicles should drive as fast as possible and on the same time they need to keep as close as possible to each other without creating or being involved in a crash (accident), for the whole duration of the test which is 3minutes. At the end of the 3 minutes the 4 vehicles will have 1 minute to park in the space without touching. An additional score will be given for each correctly parked vehicle. The parking area will be indicated by a green background. Each team has 10 minutes available to position the vehicles, start the test, complete it and park the vehicles). Teams that do not manage to finish, won't be scored. The scoring of the teams that will finish depends on an automated camera-based system and the penalties based on the ineligible behavior as described in the "Scoring and Evaluation" section.

Urban Scenario (#-shape track)

The Urban scenario takes place on the #-shape track. Each team sets up all four vehicles in the four different initial positions of the track. Each starting position can have any of the available colours. The color is assigned to the vehicle. At their starting position, the vehicles must be positioned with both wheels on a portion of the black track; only the sensors (extra 3 cm) can be on the white background. The car should be able to read and memorize its colour in the beginning of the race. During the race, the colour will not change. The vehicle should be able to recognize it through its sensors. When an action from the vehicle is expected (turn left or right) there will be a traffic sign with the corresponding color at the intersection. There will be signs only when the vehicle has to turn. If not, the car has to go straight. The vehicles should follow as quickly as possible the paths indicated by the signs of their assigned color without creating or being involved in a crash (accident), for the whole duration of the test which is 3minutes. Each team has 10 minutes available to position the vehicles, start the test and complete it. The scoring of the teams that will finish depends on the automated camera referee and the penalties based on the ineligible behavior as described in the "Scoring and Evaluation" section.

Please note:

This is a one-lane track. Each road will have a unique direction. Vehicles can meet only in the intersections.

The material will be chosen in order not to create reflections of light, probably fabric or matte paper.

The order of the cars be selected in a random way.

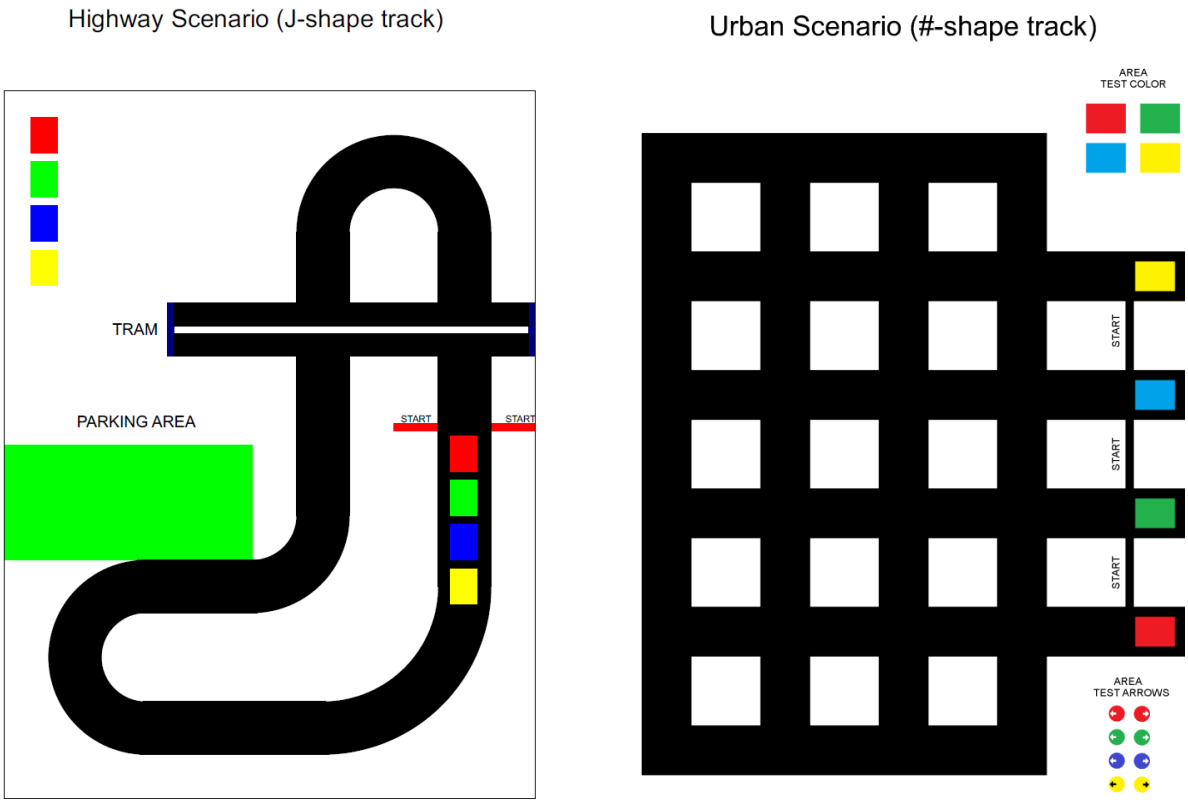


Fig. 1. Networks layouts for the AUTOTRAC Competition. On the left the J-shape track corresponding to highway conditions. On the right the #-shape track corresponding to urban conditions.

Mixed tests (J-shape track)

After all teams will have completed their tests, they will be randomly divided in 4 groups. Per each group, each team will made available one or more vehicles. The leading vehicle will be chosen randomly. The four newly formed fleets of four vehicles will repeat exactly the same tests described in the previous section for the highway scenario for the whole duration of the demonstration test which is 3 minutes. In these tests the vehicles will not be able to cooperate but still they have to prove their capability to move safely and efficiently. The objective of these tests is to show the effect of the lack of cooperation on safety and driving efficiency.

Please note that mixed test is only for demonstration purposes. It will not affect the assessment procedure. The teams can chose to participate with any car they want. For this demonstration competition, it will be possible to modify the vehicle's hardware and software.

7. Traffic signs

There will be two types of traffic signs in the Urban Scenario to signal vehicles to turn right or turn left. **If there is no sign at an intersection the car has to go straight if this possible, otherwise they have to turn left**. All signs are made in 8 cm diameter as shown in Fig. 2. They will be placed in the proximity of some intersection, on the right of the lane, at a height of 10cm from the ground as shown in Fig.2. Three signs will

have a color background (Red, Green, Blue), and a White arrow in the middle to indicate three different directions. One sign will have a Yellow background and a Black arrow.

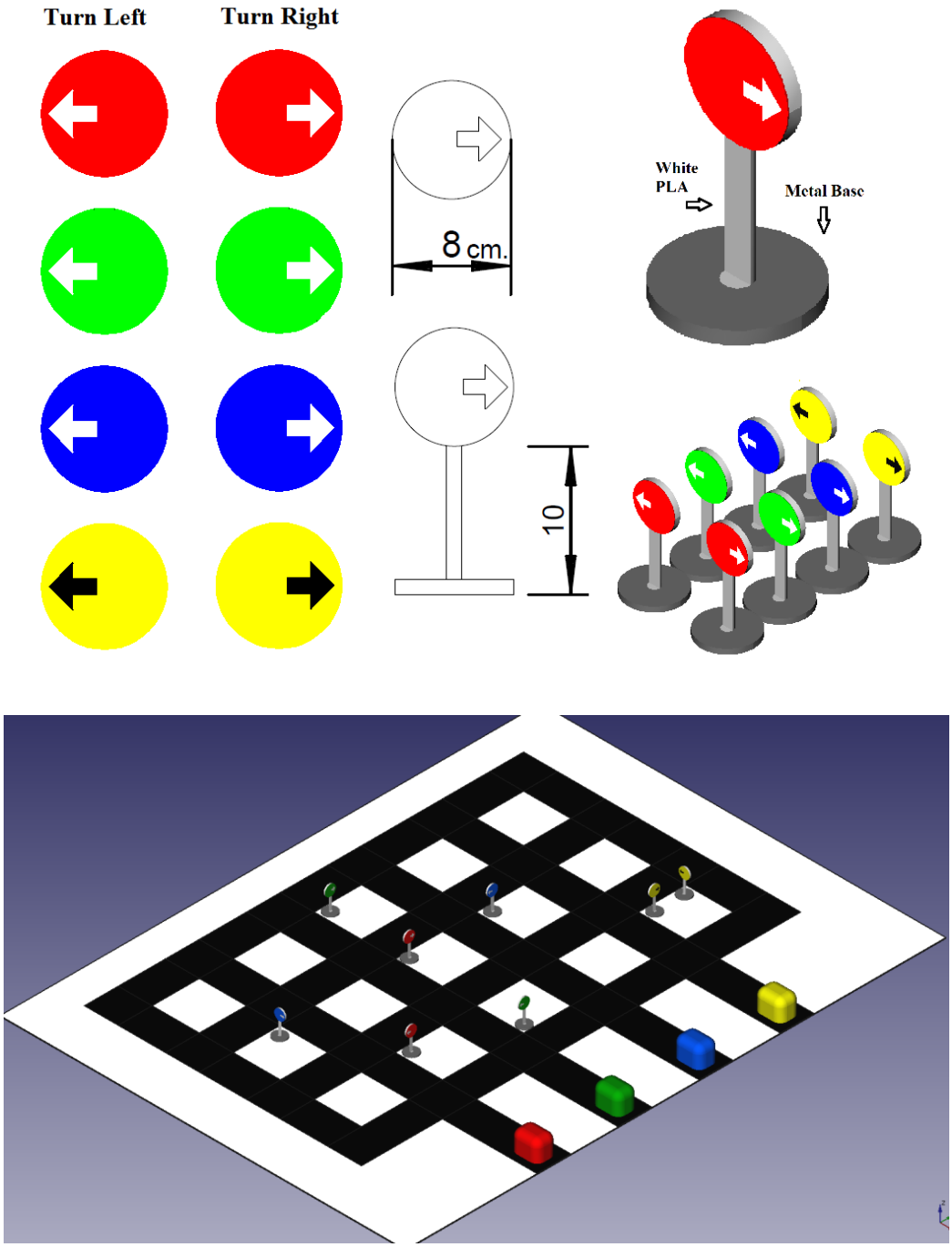


Fig. 2 The traffic signs.

8. Scoring and Evaluation

The scoring for each team that participates in the competition will be comprised by the aggregated points between different metrics. The evaluation will be performed by a group of individuals referee experts (IRE) and by an automated camera-based referee system (ACRS).

The performance of each team will be assessed based on three factors:

1. **Documentation (mandatory)**
2. **Presentation (mandatory)**
3. **Participation**

Documentation: Each team will have to deliver documentation before the event. The documentation is a prerequisite for participation. It should describe with clarity the vehicle sensors, the algorithms used for environmental perception, and any implemented control logic or data fusion process used. *The final documentation should be no more than 10 pages of MS Word and should be submitted by the day of the competition to the JRC.*

Presentation: Each team will have to present their participation (fleet of AVs, control logics, development stages, validation, sensors, data fusion, machine learning etc) in the JRC during the first day.

Participation: The assessment of each team will be based on the following metrics:

- Collision: Each vehicle that manages to finish without collision will be awarded with 25 points. Each collision costs 5 points. No points are awarded with more than four collisions.
- Lane keeping: Each vehicle that manages to stay within the lanes and finish will be awarded with 25 points. A vehicle will be considered out of lanes when at least two wheels are outside the route. Every time a vehicle gets out of lanes will lose 5 points. No points are awarded for more than four lane keeping infringement.
- Parking (only for highway track): For the Highway scenario each vehicle parked correctly will be awarded with 25 points. A correct parking means no touch with other vehicles and outside the main road lane.
- Time gap (only for highway track): For each of the three following vehicles, the ACRS will record their instantaneous time gaps with a regular frequency and after the successful completion of the task by the team, it will report the following value:

$$TG_j = \sum_{i=1}^{3} \{tg_{med} + tg_{perc90} + tg_{perc70}\}_{i,j}$$

  where tg_{med} is the median time gap and tg_{perc90} and tg_{perc70} are the 90th the 70th percentiles in the time gap values distribution, i the number of the vehicle and j the participating team. The final score will be computed as follows:

$$\widehat{TG_j} = 1 - \frac{(TG_j - TG_{min})}{TG_{max} - TG_{min}}$$

  where $TG_{max} and\ TG_{min}$ are the minimum and maximum time gap values computed from all the teams that managed to complete the challenge. The final score $\widehat{TG_j}$ will be in the range $[0,1]$ and the total points for the team *j* are determined by the product of $\widehat{TG_j} * 100$, a value in the range $[0,100]$. Teams that do not manage to complete the challenge (all vehicles should complete their race) get no points for this challenge.
- Distance travelled: For each of the vehicles, the ACRS will estimated their total distance travelled and if the vehicle finishes it will report the estimated distance travelled.

$$DT_j = \sum_{i=1}^{3} \{DT\}_{i,j}$$

  where DT_j is the accumulated distance traveled by all vehicles *i* of the team *j*. The final score will be computed as follows:

$$\widehat{DT_j} = \frac{DT_j - DT_{min}}{DT_{max} - DT_{min}}$$

  where $DT_{max}\ and\ DT_{min}$ are the minimum and maximum distance travelled values computed between the teams that managed to complete the challenge. The final score $\widehat{DT_j}$ will be in the range $[0,1]$ and the total points for the team *j* are determined by the product of $\widehat{DT_j} * 100$, a value in the

range $[0,100]$. Teams that do not manage to complete the challenge (all vehicles should complete their race) get no points for this challenge.

The final score for each team will be computed by aggregation of the individual metric scores.

Table I. Assessment table (scoring)

	Metric	Maximum points
Highway / Urban Scenario	Collision	100
	Lane keeping	100
	Distance travelled	100
Only highway Scenario	Time gap	100
	Parking	100

9. Next steps and Competition

Qualification

Each team selected for participation in the JRC AUTOTRAC should provide the following by the **15th of February 2020**:

- **A short video file (not more than 30s) with the vehicles and a 1-page document describing the development updates.**

Competition

The competition will last 3 days. The first day (warm up), the JRC will present the EU activities on the topic and each team will present its proposed solution. During the afternoon, teams will be able to inspect the space where the competition with take place and the tracks. The second and third day, the teams will compete in the AUTOTRAC 2020 on the J-shape and #-shape tracks respectively. The participation order will be random and extracted on the day of the competition. The team that will participate in the **J-shape track** first, will be the last of the **#-shape track** and so on.

A buffer of 30min plus 10min for competition will be given to each participating team. This practically means that the teams will have the time to prepare their vehicles on the track. When a team is ready or the latest after 30mins, the team will have to compete. Failure to successfully complete the challenge for any team will mean zero points in the evaluation procedure.

Further information will be communicated to the teams **only via email** as we approach the date of the competition. If you have any questions, please send them to the JRC-AUTOTRAC@ec.europa.eu

Indicative agenda – JRC-AUTOTRAC 2020

1st day	**2nd day**	**3rd day**
8:00 – 9:00 Welcome – Registration 9:00 – 10:00 JRC Welcome and presentation of the main activities of the European Commission. 10:00 - 13:00 Presentations of the participants13:00 – 14:30 – Lunch – 14:30 – 17:30 – Free vehicle testing 18:30 – 21:00 – Buffet/Dinner	8:00 – 9:00 Welcome 9:00 - 13:00 **Competition (#-track) 30mins buffer per team + 10mins performance.** 13:00 – 14:30 – Lunch – Technical Break 14:30 – 16:00 – **Competition continuation** **16:00-17:00 Presentation of preliminary results on the J-shape track.**	**8:00 – 9:00 Welcome** **9:00 - 13:00 Competition (#-track) 30mins buffer per team + 10mins performance.** **13:00 – 14:30 – Lunch – Technical Break** **14:30 – 16:00 – Sharing/Networking (& scoring)** **16:00 – 17:00 – Awards – (See you next year)**

Revision history

10/05/2019 – Rev.01- First version

07/11/2019 – Rev.02

Please read the second version carefully as it is updated on several parts of the document. Interesting updates include:

- The competition is organized by the Joint Research Center (JRC) of the European Commission and it will take place in Ispra (VA) from March 30 to April 1 2020.
- The document has been updated with the Scoring and Assessment procedure.
- The first, second and third team will be awarded with the amount of 3.000€, 2.000€, 1.000€ respectively. Furthermore, for the winning team of the challenge there will be the possibility to develop a demonstrator in the Ispra site of the JRC with a contract of around 15.000€. Finally, the winning team will have the opportunity to demonstrate their performance during the 2020 Transportation Research Arena conference, which will be held on April 27-30, 2020 in Helsinki (https://traconference.eu/).
- The JRC will provide fixed financial support for travel and accommodation for all the participated teams.
- The vehicle's maximum weight should be 3kg.
- Each vehicle must be equipped with at least one camera for environment recognition.
- It is possible to use smartphones or tablets (not PCs) beyond the maximum measurements and weight of the rules.
- During the competition (3nd day) the hardware of the vehicle must not be modified except in case of supervised repair.
- The software can be changed during the competition (3nd day), but not during the run
- Each team is free to apply V2V connectivity between its vehicles. No WiFi network for such reason will be provided by the JRC.
- Update GREEN color CMYK: 100, 0, 100, 0 RGB: 0, 255, 0
- Updated: Urban Scenario (#-shape track)
- Mixed event is only for demo purposes. It will not affect the assessment procedure. The teams can chose to participate with any car they want.
- The technical specifications will freeze one month before the competition. From the moment that the list of participant is fixed, any updates will be communicated via email and not via the website.
- Up to 4 members of each team are allowed to attend the competition in the Ispra premises of the JRC. However, there is no restriction the individuals that contribute in the preparation of the team.
- There will be signs only when the vehicle has to turn. If not, the car has to go straight.
- Update the colors of the yellow / black signal

- The car should be able to read and memorize its colour in the beginning of the race. During the race, the colour will not change.
- Mixed test is only for demonstration purposes

JRC AUTOTRAC 2020 competition: Technical document of Team TU Delft

Student team: Jelle Baltus, Xander van der Geest, Jurriaan Govers, Max van der Linden, Anand Padmanabhan, Cas Rosier; **Supervisor team:** Suad Krilasevic, Guhao Sun (PhD students); Sergio Grammatico (Associate Prof.)

Laboratory setup:

Our laboratory setup for testing consists of two (1:1 scale) tracks printed on a PVC banner, one for the “J-shaped” track and one for the “urban scenario” track. As robots, we use Husarion’s ROSbot 2.0, connected to a separate router. This enables us to read out the sensors and upload C++ ROS code to each robot.

Highway-scenario:

We first developed a lane keeping algorithm that works by first taking the camera image and denoising it with a Gaussian filter. Next, a conversion into grayscale is done since colours are not relevant for lane keeping. The edges of the lane are determined by looking at the contrast and using a Sobel edge detection with an adaptive threshold. Once the edges are found, a Hough transformation is used to determine the sidelines of the lane. To stay in the middle of the lane, we designed a PI-controller that adjusts the rotation rate of the ROSbot. The forward velocity of the platoon is determined by the leader vehicle. The following vehicles have a slightly higher speed to catch the platoon, should they be separated. Two distance sensors on the front of the ROSbots are used to avoid collisions. Moreover, a P-controller on the measured distance slows the approaching vehicle down, for smoothening the overall platoon.

Tram detection:

To detect the tram, we used 2D lidar scan data. In particular, the speed and direction of the tram are estimated, to then ensure safe passage of the platoon. Specifically, the first robot calculates if it is possible for the last robot to pass; if not, all the robots wait until the tram has passed.

Parking:

After the three minutes, a timer runs out, so we let the robots use an HSV filter to detect the green parking area. If the number of green pixels hits a certain threshold, then the robot is standing next to the parking area, so it can start the parking maneuver. The latter is controlled via the odometry data, from which an initial position and orientation is stored. These are then used to compute relative distance and orientation with respect to a goal position, such that a P-controller can move the robot to its final parking position.

Urban-scenario:

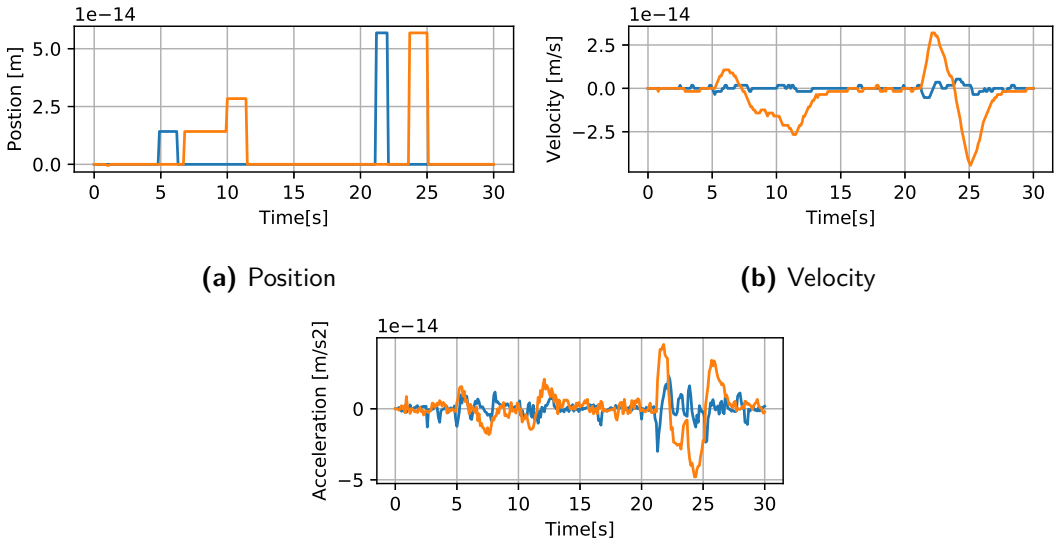
The lane keeping algorithm for the urban scenario relies on the same fundamentals as those for the highway scenario. However, we extended the algorithm further to also detect approaching junctions while the robot is driving forward. If the robot is within a certain proximity of a junction, then a different controller is activated to conduct an appropriate maneuver. In addition, we use a global tracking algorithm to determine what specific junction the robot is approaching. This is communicated to the rest of the robots and is used together with the lane keeping algorithm to determine the “right of way” on a junction. The sign detection works on two levels: detecting “objects” that are circular in shape, and then identifying the color and the shape of the arrow on it. Based on the detections, we transmit two values that are used by the controller to make decisions and in turn command the appropriate maneuver.

YouTube channel: https://www.youtube.com/channel/UCYuZkyMt0O1GAKoYvcPUu2A

30-second qualification video: https://www.youtube.com/watch?v=3xsG7R2bY5o

Appendix B

Numerical differences

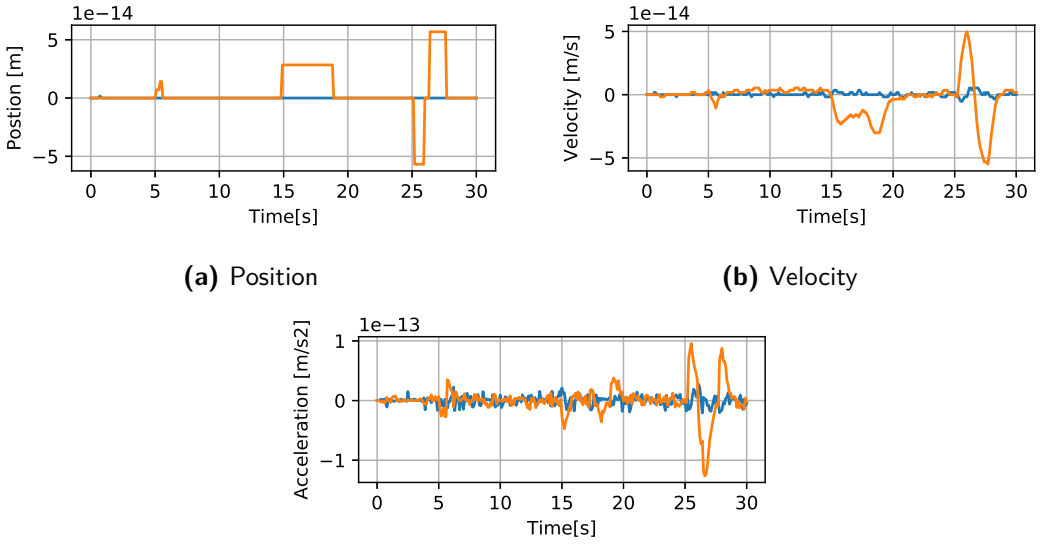


(a) Position

(b) Velocity

(c) Acceleration

Figure B-1: Numerical differences between plaintext and encrypted implementation of a simulation with two agents



(a) Position

(b) Velocity

(c) Acceleration

Figure B-2: Numerical differences between encrypted version and plaintext version of simulation with four agents

Appendix C

Additional simulation results

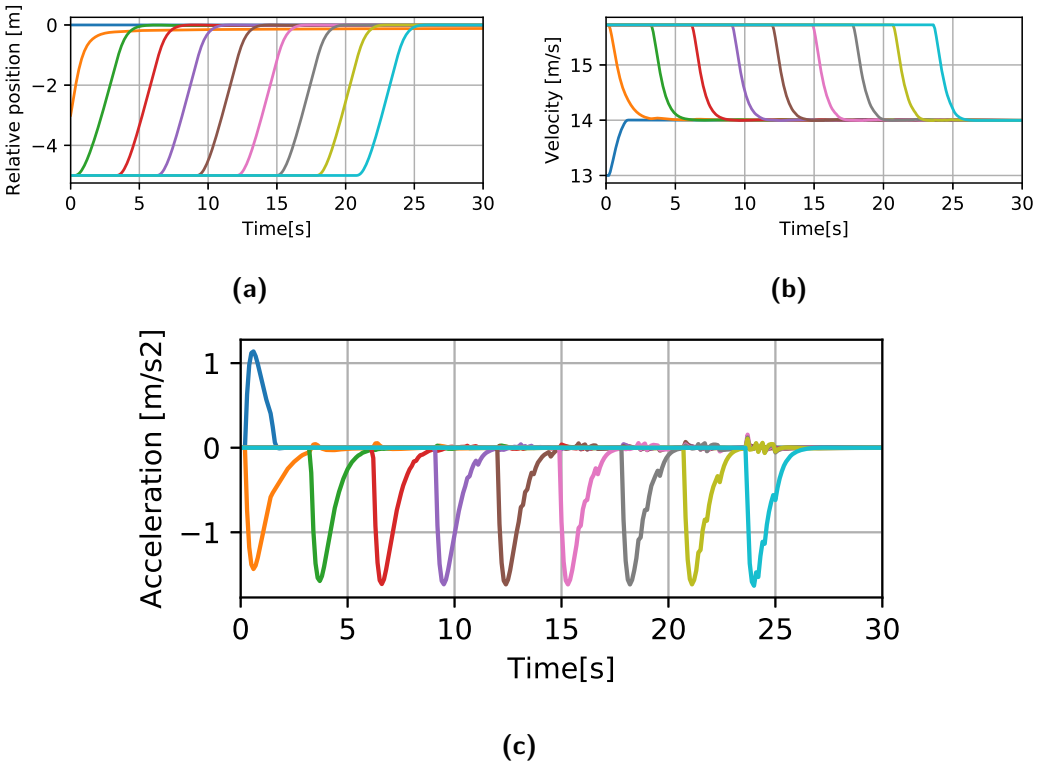
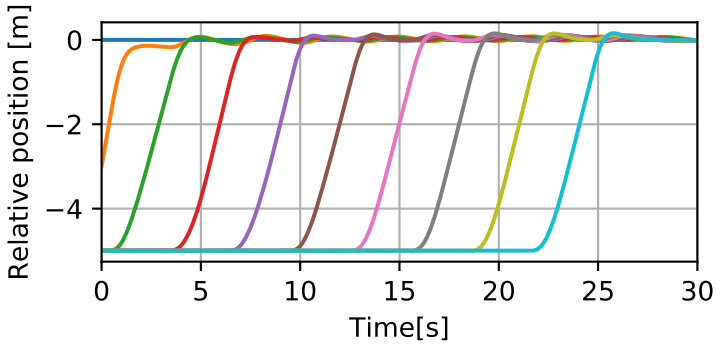
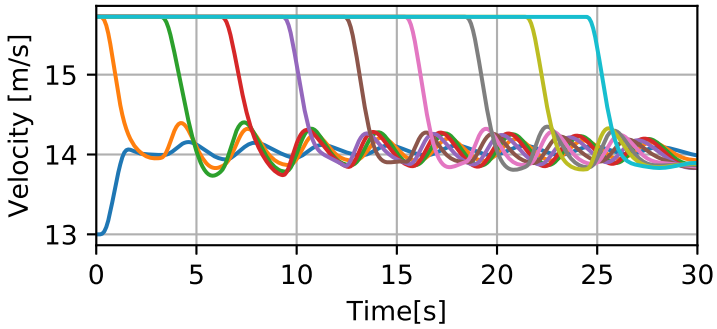


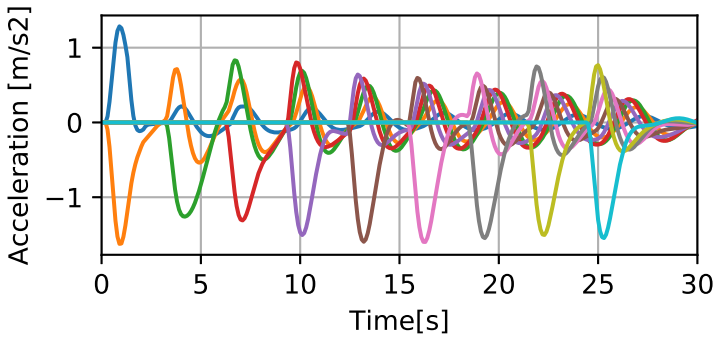
Figure C-1: Results of a simulation in which the trajectories of ten vehicles are coordinated by the PMPC. Fig. (a) shows the positions of the vehicles, relative to the safety distance of its neighbor. Fig. (b) shows the velocities of the vehicles, the leader-vehicle starts at 13 [m/s] and the follower-vehicles at 15.725 [m/s]. The velocity of the leader-vehicle is constrained at 14 [m/s]. Fig. (c) shows the acceleration, the control input of the vehicles.



(a)



(b)



(c)

Figure C-2: Results of a simulation in which the trajectories of ten vehicles are coordinated by the PMPC, and only a single optimization iteration is executed each time-step. Fig. (a) shows the positions of the vehicles, relative to the safety distance of its neighbor. Fig. (b) shows the velocities of the vehicles, the leader-vehicle starts at 13 [m/s] and the follower-vehicles at 15.725 [m/s]. The velocity of the leader-vehicle is constrained at 14 [m/s]. Fig. (c) shows the acceleration, the control input of the vehicles.

Bibliography

[1] J. Alonso-Mora, M. Rufli, R. Siegwart, and P. Beardsley, "Collision avoidance for multiple agents with joint utility maximization," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2833–2838, IEEE, 2013.

[2] W. Schwarting, J. Alonso-Mora, L. Pauli, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1928–1935, 2017.

[3] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Coordination of Multiple Vessels Via Distributed Nonlinear Model Predictive Control," in *2018 European Control Conference (ECC)*, pp. 2523–2528, European Control Association (EUCA), 2018.

[4] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[5] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments," *Autonomous Robots*, pp. 1–22, 2018.

[6] X. Qian, A. de La Fortelle, and F. Moutarde, "A hierarchical Model Predictive Control Framework for On-road Formation Control of Autonomous Vehicles," in *2016 IEEE Intelligent Vehicles Symposium*, (Göteborg, Sweden), pp. 376–381, 2016.

[7] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G. J. Balas, "Decentralized Receding Horizon Control and Coordination of Autonomous Vehicle Formations," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 1, pp. 19–33, 2008.

[8] S. K. Mulagaleti, R. Van Parys, and G. Pipeleers, "Distributed model predictive control of multiple vehicles transporting a flexible payload," in *2018 European Control Conference (ECC)*, pp. 1417–1422, European Control Association (EUCA), 2018.

[9] Y. Lin, F. Farokhi, I. Shames, and D. Nesic, "Secure Control of Nonlinear Systems Using Semi-Homomorphic Encryption," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 5002–5007, 2018.

[10] F. Farokhi, I. Shames, and N. Batterham, "Secure and Private Cloud-Based Control Using Semi-Homomorphic Encryption," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 163–168, 2016.

[11] F. Farokhi, I. Shames, and N. Batterham, "Secure and private control using semi-homomorphic encryption," *Control Engineering Practice*, vol. 67, no. August, pp. 13–20, 2017.

[12] M. Schulze Darup, A. Redder, and D. E. Quevedo, "Encrypted Cooperative Control Based on Structured Feedback," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 37–42, 2019.

[13] M. Schulze Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted MPC for linear constrained systems," *IEEE Control Systems Letters*, vol. 2, no. 2, pp. 195–200, 2018.

[14] A. B. Alexandru, M. Morari, and G. J. Pappas, "Cloud-Based MPC with Encrypted Data," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 5014–5019, 2018.

[15] M. Schulze Darup, A. Redder, and D. E. Quevedo, "Encrypted cloud-based MPC for linear systems with input constraints," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 535–542, 2018.

[16] M. Schulze Darup, *Encrypted Model Predictive Control in the Cloud*. Springer Singapore, 2020.

[17] Y. Shoukry, K. Gatsis, A. Alanwar, G. J. Pappas, S. A. Seshia, M. Srivastava, and P. Tabuada, "Privacy-Aware Quadratic Optimization Using Partially Homomorphic Encryption," in *2016 IEEE Conference on Decision and Control (CDC)*, pp. 5053–5058, 2016.

[18] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2 ed., 2009.

[19] E. F. Camacho and C. Bordons, *Model Predictive Control*. Springer International Publishing, 2 ed., 2003.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 7 ed., 2004.

[21] I. Necoara, L. Ferranti, and T. Keviczky, "An adaptive constraint tightening approach to linear model predictive control based on approximation algorithms for optimization," *Optimal Control Applications and Methods*, vol. 36, pp. 648–666, 2015.

[22] F. Armknecht, C. Boyd, C. Carr, K. Gjosteen, A. Jaschke, C. A. Reuter, and M. Strand, "A Guide to Fully Homomorphic Encryption," tech. rep., 2015.

[23] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[24] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[25] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," *Eurocrypt*, vol. 1592, pp. 223–238, 1999.

[26] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. CRC Press, 2007.

[27] E. Barker, L. Chen, A. Roginsky, A. Vassilev, R. Davis, and S. Simon, "Recommendation for Pair-Wise Key- Establishment Schemes Using Integer Factorization Cryptography (Revision 2)," *NIST Special Publication 800-56B*, 2019.

[28] C. Jost, H. Lam, A. Maximov, and B. J. Smeets, "Encryption Performance Improvements of the Paillier Cryptosystem," *IACR Cryptology ePrint Archive*, 2015.

[29] C. Data61, "Python paillier library." https://github.com/data61/python-paillier, 2013.

[30] D. Thanou, E. Kokiopoulou, Y. Pu, and P. Frossard, "Distributed Average Consensus With Quantization Refinement," *IEEE Transactions on Signal Processing*, vol. 61, no. 1, pp. 194–205, 2013.

[31] T. Veugen, F. Blom, S. J. De Hoogh, and Z. Erkin, "Secure comparison protocols in the semi-honest model," *IEEE Journal on Selected Topics in Signal Processing*, vol. 9, pp. 1217–1228, oct 2015.

[32] M. Nateghizad, Z. Erkin, and R. L. Lagendijk, "An efficient privacy-preserving comparison protocol in smart metering systems," *Eurasip Journal on Information Security*, vol. 2016, no. 1, 2016.

[33] D. Evans, V. Kolesnikov, and M. Rosulek, *A Pragmatic Introduction to Secure Multi-Party Computation*. NOW Publishers, 2018.

[34] A. B. Alexandru and G. J. Pappas, "Secure Multi-party Computation for Cloud-Based Control," *Privacy in Dynamical Systems*, pp. 179–207, 2020.

[35] W. Zheng, R. A. Popa, J. E. Gonzalez, I. Stoica, R. Ada Popa, J. E. Gonzalez, I. Stoica, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously Secure Coopetitive Learning for Linear Models," in *IEEE Symposium on Security and Privacy*, 2019.

[36] J. Tran, F. Farokhi, M. Cantoni, and I. Shames, "Implementing Homomorphic Encryption Based Secure Feedback Control for Physical Systems," *arXiv e-prints*, 2019.

Glossary

List of Acronyms

ADMM	Alternating Direction Method of Multipliers
DARE	Discrete-time Algebraic Riccati Equation
FHE	Fully Homomorphic Encryption
HE	Homomorphic Encryption
JRC	Joint Research Centre
MPC	Model Predictive Control
PGS	Projected Gradient Scheme
PHE	Partially Homomorphic Encryption
PMPC	Privacy-preserving Model Predictive Controller
QP	Quadratic Programming
SMPC	Secure Multi-Party Computation
SQP	Sequential Quadratic Programming

List of Symbols

δ	Maximum allowed constraint violation
η	Step-size of the Projected Gradient Scheme
μ	Dual variables
∇	Gradient
$\phi(.)$	Euler's totient function
ε	Stopping criterion
$[\![.]\!]$	Encryption
$\mathbb{N}$	Natural numbers, $\mathbb{N} = \mathbb{Z}_{\geq 0}$

$\mathbb{R}$	Real numbers
$\mathbb{Z}$	Integers
$\mathbb{Z}_M$	Additive group moduluo M
$\mathbb{Z}_M^*$	Multiplicative group modulo M
$\mathcal{D}(.)$	Decryption
a	Acceleration
A, B, C, D	System matrices
d_{safe}	Safety distance
e	State constraint vector
$\bar{e}$	State evolution constraint vector
$e_{c,ij}$	Coupling state constraint vector between vehicles i and j
$\bar{e}_{c,ij}$	Coupling state evolution constraint vector between vehicles i and j
$\boldsymbol{e_x}$	State evolution constraint vector of multiple vehicles
$\boldsymbol{e}$	Input sequence constraint vector of multiple vehicles
E	State constraint matrix
$\bar{E}$	State evolution constraint matrix
$E_{c,ij}$	Coupling state constraint matrix of vehicle i with vehicle j
$\bar{E}_{c,ij}$	Coupling state evolution constraint matrix of vehicle i with vehicle j
$\boldsymbol{E_x}$	State evolution constraint matrix of multiple vehicles
$\boldsymbol{E}$	Input sequence constraint matrix of multiple vehicles
$\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}$	Cost function matrices
$g(.)$	Dual function
k	Current time-step
$\ell(.)$	Stage cost function
$L(.)$	Lagrangian
M	Public key, hard-to-factor number consisting of two large primes
m	Number of inputs
N	Prediction horizon
n	Number of states
p	Position
P, S	Prediction matrices
$\boldsymbol{P}, \boldsymbol{S}$	Prediction matrices of multiple vehicles
Q	State weight matrix
Q^f	Final state weight matrix
$\bar{Q}$	State evolution weight matrix
$\bar{\boldsymbol{Q}}$	State evolution weight matrices of multiple vehicles
R	Input weight matrix
$\bar{R}$	Input sequence weight matrix
$\bar{\boldsymbol{R}}$	Input sequence weight matrices of multiple vehicles
t	Sampling-time

u	Input
$\bar{u}$	Input sequence, $(u(0), u(1), \ldots, u(N-1))$
$\bar{\boldsymbol{u}}$	Input sequences of multiple vehicles
V	Number of vehicles
v	Velocity
v_{follow}	Velocity set-point of the follower-vehicles
v_{lead}	Velocity set-point of the leader-vehicle
v_{platoon}	Maximum velocity of the platoon
$\mathcal{V}(.)$	Cost function
$\mathcal{V}_f(.)$	Final cost function
x	State
x_0	Current state
$\boldsymbol{x_0}$	Current states of multiple vehicles
$\tilde{\boldsymbol{x}}_{\boldsymbol{0}}$	Translated current states of multiple vehicles
$\bar{x}$	State evolution, $(x(0), x(1), \ldots, x(N))$
$\bar{\boldsymbol{x}}$	State evolution of multiple vehicles
y	Output
$_{\geq 0}$	Group of positive values, containing zero
$_{i,j}$	Vehicle i or j
*	Optimal solution
$^+$	Value used in next optimization iteration