Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 53-2 (2020) 3508–3514

Encrypted MPC based on ADMM real-time iterations*

Moritz Schulze Darup

Encrypted Control Group, Universität Paderborn, Germany.
(e-mail: moritz.schulzedarup@rub.de)

Abstract: Encrypted control enables confidential controller evaluations in cloud-based or networked control systems. Technically, an encrypted controller is a modified control algorithm that is capable of computing encrypted control actions based on encrypted system states without intermediate decryption. The realization of such controllers, e.g., using homomorphic encryption, is non-trivial. Nevertheless, even optimization-based model predictive control (MPC) has already been implemented in an encrypted fashion. However, the existing schemes either require an explicit solution of the parametric optimal control problem (OCP) or they can only consider input constraints. In this paper, we present a novel encrypted MPC that allows to include state and input constraints without the requirement of an explicit solution of the OCP. The approach builds on the encrypted implementation of a single iteration of the alternating direction method of multipliers (ADMM) per time step, i.e., ADMM real-time iterations.

Keywords: Encrypted control, model predictive control (MPC), alternating direction method of multipliers (ADMM), real-time iterations, homomorphic encryption

1. INTRODUCTION

Cloud-computing and distributed computing are becoming omnipresent in modern control systems such as smart grids, robot swarms, building automation, or intelligent transportation systems. While cloud-based and distributed control schemes usually increase the systems' scalability and performance, these schemes also raise the risk of cyberattacks. In fact, the required communication and processing of sensitive data via public networks and on third-party platforms promote interception and manipulation of data. Future control schemes should counteract those threats and ensure confidentiality, integrity, and availability of the involved data.

A promising setup for cloud-based control that effectively prevents the risk of eavesdropping is illustrated in Figure 1. In this setup, that has first been proposed in Kogiso and Fujita (2015), state measurements are encrypted at the sensor and sent to the cloud. In the cloud, an encrypted version of the controller computes an encrypted control action (without intermediate decryption) and sends it to the actuator. At the actuator, the control action is decrypted and applied to the system. Confidentiality is guaranteed throughout the control loop since system states and inputs are encrypted not only during transmission but also during computations in the cloud. While the concept is straightforward, its technical realization is challenging. In principle, homomorphic encryption (see, e.g., Paillier (1999); Gentry (2009)) allows to rewrite any algorithm such that it computes encrypted outputs based on encrypted inputs. However, in practice, encrypting algorithms is currently realizable only for simple procedures since solely encrypted sums and/or multiplications can be carried out with a reasonable numerical effort. Yet, these operations already allow for the encrypted evaluation of simple control laws such as linear state feedback (see Kogiso and Fujita (2015); Kim et al. (2016); Farokhi et al. (2017)).

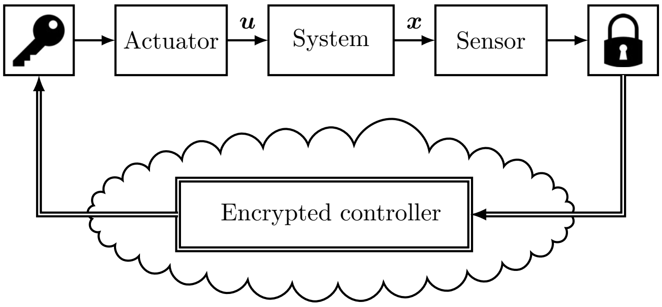

Fig. 1. Cloud-based control scheme with encrypted communications and encrypted controller evaluation.

In this paper, we focus on encrypted model predictive control (MPC). Clearly, due to optimization-based control actions and the presence of state and/or input constraints, encrypting MPC schemes is considerably harder than encrypting linear state feedback. Nevertheless, encrypted MPC has recently been realized in Alexandru et al. (2018), Schulze Darup et al. (2018a), and Schulze Darup et al. (2018b). However, the two first mentioned schemes only consider input constraints and the third approach requires an explicit solution of the multi-parametric optimal control problem (OCP). Here, we present an encrypted MPC that can handle state and input constraints and that does not require an offline solution of the OCP. The novel controller builds on an encrypted implementation of the alternating direction method of multipliers (ADMM). More precisely, encrypted real-time ADMM iterations are considered, i.e., only one solver iteration per time-step.

Real-time iterations for feedback control have been introduced in Li and Biegler (1989) and Diehl et al. (2002). At

* Support by the German Research Foundation (DFG) under the grant SCHU 2940/4-1 is gratefully acknowledged.

10.1016/j.ifacol.2020.12.1708

first sight, considering a single solver iteration per time-step seems doomed to fail. However, in the framework of control, a single iteration per sampling instant can be sufficient since further iterations follow at future time-steps. For the special case of linear-quadratic MPC, one can even certify asymptotic stability as recently shown in van Parys and Pipeleers (2018) and Schulze Darup and Book (2019b) for projected gradient and ADMM real-time iterations, respectively. Such real-time iterations are suitable for homomorphically encrypted implementations since only one projection step is involved in the controller evaluation. Encrypting these projections (in an efficient manner) is difficult. However, the single projection can be outsourced from the cloud to the actuator. This approach has been successfully applied in Schulze Darup et al. (2018a) to encrypt the projected gradient scheme from van Parys and Pipeleers (2018). In this paper, we use a similar approach to encrypt the ADMM real-time iterations considered in Schulze Darup and Book (2019b). Apart from the inclusion of state constraints, the novel scheme differs from Schulze Darup et al. (2018a) in that some encrypted data is buffered in the cloud (see Fig. 2). As detailed below, this results in a more sophisticated encrypted controller.

The paper is organized as follows. We state notation in the remainder of this section and provide background on MPC, homomorphic encryption, and (required) quantization in Section 2. The main contribution of the paper, i.e., the encrypted implementation of the predictive control scheme based on ADMM real-time iterations, is presented in Section 3. The approach is illustrated with an example in Section 4. Finally, conclusions and an outlook are given in Section 5.

Notation. We denote the sets of real, integer, and natural numbers by $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{N}$, respectively. Positive natural numbers are denoted by $\mathbb{N}_+$. The sets $\mathbb{Z}_P$ and $\mathbb{Z}_P^*$ refer to the additive and multiplicative group of integers modulo $P \in \mathbb{N}_+$, respectively, where standard representatives of $\mathbb{Z}_P$ are collected in the set $\mathbb{N}_P := \{0, 1, \dots, P-1\}$. Regarding modulo operations, we use the convention

$$z \bmod P := z - P \left\lfloor \frac{z}{P} \right\rfloor,$$

where $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ refer to the floor respectively the ceiling function. Furthermore, we write bold-face letters whenever we refer to matrices or vectors. In particular, we denote the identity matrix in $\mathbb{R}^{n\times n}$ by $\boldsymbol{I}_n$. In addition, with $\mathbf{1}_m$, $\mathbf{0}_m$, and $\mathbf{0}_{m\times n}$, we refer to the vector in $\mathbb{R}^m$ full of ones, the zero vector in $\mathbb{R}^m$, and the zero matrix in $\mathbb{R}^{m\times n}$, respectively. Moreover, for a positive definite matrix $\boldsymbol{P}$, we define $\|\boldsymbol{x}\|_{\boldsymbol{P}}^2 := \boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x}$. Finally, the indicator function of some set $\mathcal{Z} \in \mathbb{R}^p$ is defined as

$$\mathcal{I}_{\mathcal{Z}}(\boldsymbol{z}) := \begin{cases} 0 & \text{if} \quad \boldsymbol{z} \in \mathcal{Z}, \\ \infty & \text{otherwise.} \end{cases}$$

2. BACKGROUND ON MPC, HOMOMORPHIC ENCRYPTION, AND QUANTIZATION

2.1 Model predictive control for linear constrained systems

We consider linear discrete-time systems

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \qquad \boldsymbol{x}(0) := \boldsymbol{x}_0, \tag{1}$$

with state and input constraints of the form

$$\boldsymbol{x}(k) \in \mathcal{X} \subset \mathbb{R}^n \qquad \text{and} \qquad \boldsymbol{u}(k) \in \mathcal{U} \subset \mathbb{R}^m.$$

For this system class, MPC builds on solving the OCP

$$\begin{aligned} V(\boldsymbol{x}) := \min_{\substack{\tilde{\boldsymbol{x}}(0),\dots,\tilde{\boldsymbol{x}}(N) \\ \tilde{\boldsymbol{u}}(0),\dots,\tilde{\boldsymbol{u}}(N-1)}} \; & \|\tilde{\boldsymbol{x}}(N)\|_{\boldsymbol{P}}^2 + \sum_{\kappa=0}^{N-1} \|\tilde{\boldsymbol{x}}(\kappa)\|_{\boldsymbol{Q}}^2 + \|\tilde{\boldsymbol{u}}(\kappa)\|_{\boldsymbol{R}}^2 \\ \text{s.t.} \qquad & \tilde{\boldsymbol{x}}(0) = \boldsymbol{x}, \\ & \tilde{\boldsymbol{x}}(\kappa+1) = \boldsymbol{A}\,\tilde{\boldsymbol{x}}(\kappa) + \boldsymbol{B}\tilde{\boldsymbol{u}}(\kappa), \qquad \forall \kappa \in \mathbb{N}_N \\ & \tilde{\boldsymbol{x}}(\kappa) \in \mathcal{X}, \qquad \forall \kappa \in \mathbb{N}_N \\ & \tilde{\boldsymbol{u}}(\kappa) \in \mathcal{U}, \qquad \forall \kappa \in \mathbb{N}_N \\ & \tilde{\boldsymbol{x}}(N) \in \mathcal{T} \end{aligned} \tag{2}$$

in every time step for the current state $\boldsymbol{x} = \boldsymbol{x}(k)$. In this context, $N \in \mathbb{N}_+$ refers to the prediction horizon, $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are weighting matrices, and $\mathcal{T} \subseteq \mathcal{X}$ is a terminal set. Throughout the paper, we make the following assumptions:

Assumption 1. The pair $(\boldsymbol{A}, \boldsymbol{B})$ is stabilizable, the matrices $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are positive definite, and $\mathcal{X}, \mathcal{U}$, and $\mathcal{T}$ are convex and closed sets with the origin as an interior point.

Assumption 2. Projections onto the sets $\mathcal{X}, \mathcal{U}$, and $\mathcal{T}$ can be efficiently computed.

While Assumption 1 is common, Assumption 2 is non-standard (and slightly imprecise). It is, e.g., satisfied for box-shaped or ellipsoidal constraints. It remains to specify the control action of an MPC, which corresponds to the first element of the optimal control sequence, i.e.,

$$\boldsymbol{u}(k) = \tilde{\boldsymbol{u}}^*(0), \tag{3}$$

where $\tilde{\boldsymbol{u}}^*(0), \dots, \tilde{\boldsymbol{u}}^*(N-1), \tilde{\boldsymbol{x}}^*(0), \dots, \tilde{\boldsymbol{x}}^*(N)$ refer to the optimizers for (2).

2.2 Homomorphic encryption and the Paillier cryptosystem

In general, homomorphic encryption (HE) refers to a special family of cryptosystems that enables certain mathematical operations to be carried out on encrypted data. More precisely, we call a cryptosystem additively homomorphic if there exists an operation "$\oplus$" such that

$$z_1 + z_2 = \texttt{Dec}\left(\texttt{Enc}(z_1) \oplus \texttt{Enc}(z_2)\right) \tag{4}$$

holds, where z_1 and z_2 are two arbitrary numbers in the message space of the cryptosystem and where the functions `Enc` and `Dec` refer to the encryption and decryption procedure, respectively. Multiplicatively HE is defined analogously. A popular additively HE scheme, that is detailed in the following, is Paillier (1999). Multiplicatively HE is often implemented using ElGamal (1985). Encryption schemes that are both additively and multiplicatively homomorphic are called fully homomorphic. In principle, fully HE can be used to encrypt arbitrary functions (Gentry, 2010). However, it is computationally highly demanding and currently not a competitive option for encrypted control. Nevertheless, so-called somewhat or leveled fully HE schemes (see, e.g., Brakerski et al. (2014)), that support a limited number of encrypted multiplications and additions, may be useful for future encrypted controllers.

Most existing encrypted controllers (such as Kim et al. (2016); Farokhi et al. (2017); Alexandru et al. (2018); Schulze Darup et al. (2018b,a); Alexandru and Pappas (2019)) make use of the Paillier cryptosystem. In this asymmetric encryption scheme, the encryption is carried out based on a public key P and the decryption requires the secret key S. The key generation for the simplified scheme (Katz and Lindell, 2014, Sect. 13.2) builds on two primes $p_1, p_2 \in [2^{\ell-1}, 2^{\ell}-1]$ of the same "length" $\ell \in \mathbb{N}_+$, where l should currently be at least 1024 for security reasons (National Institute of Standards and Technology, 2016). The public key then is $P = p_1 p_2$ and the private key evaluates to $S = (p_1-1)(p_2-1)$. The encryption of a number z from the message space $\mathbb{Z}_P$ is realized by

$$\texttt{Enc}(z,r) := (P+1)^z r^P \bmod P^2, \tag{5}$$

where r is randomly chosen from $\mathbb{Z}_P^*$ for every single encryption. The resulting ciphertext c lies in the set $\mathbb{Z}_{P^2}^*$. The decryption is carried out by computing

$$\texttt{Dec}(c) := \frac{(c^S \bmod P^2)-1}{P}\, S^{-1} \bmod P, \tag{6}$$

where $S^{-1} \bmod P$ refers to the multiplicative inverse of S modulo P. In the following, we will restrict our attention to messages from the set $\mathbb{N}_P$. We then obtain $\texttt{Dec}(\texttt{Enc}(z,r)) = z$ for every $z \in \mathbb{N}_P$ and every $r \in \mathbb{Z}_P^*$, i.e., reversibility of the encryption. The Paillier cryptosystem is additively homomorphic. In fact, we have

$$z_1 + z_2 = \texttt{Dec}\left(\texttt{Enc}(z_1,r_1)\,\texttt{Enc}(z_2,r_2) \bmod P^2\right). \tag{7}$$

for all $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathbb{N}_P$ such that $\boldsymbol{z}_1 + \boldsymbol{z}_2 \in \mathbb{N}_P$ and all $r_1, r_2 \in \mathbb{Z}_P^*$. In addition, the Paillier cryptosystems supports multiplications with one encrypted factor as apparent from the relation

$$z_1 z_2 = \texttt{Dec}\left(\texttt{Enc}(z_1,r)^{z_2} \bmod P^2\right) \tag{8}$$

that holds for all $z_1, z_2 \in \mathbb{N}_P$ such that $z_1 z_2 \in \mathbb{N}_P$ and all $r \in \mathbb{Z}_P^*$. The relations (7) and (8) form the basis of many existing encrypted controllers and they will also be essential for the novel controller proposed in Section 3.

2.3 Quantization via fixed-point numbers

The previously presented Paillier cryptosystem is designed for integer messages. Hence, applying Paillier encryption to realize encrypted control requires to map the system states and the controller parameters onto a subset of $\mathbb{Z}$. This mapping usually starts with a quantization. Here, we approximate the states $\boldsymbol{x}_j \in \mathbb{R}$ (and controller parameters) with fixed-point numbers from the set

$$\mathbb{Q}_{\beta,\gamma,\delta} := \left\{-\beta^\gamma, -\beta^\gamma + \beta^{-\delta}, \dots, \beta^\gamma - 2\beta^{-\delta}, \beta^\gamma - \beta^{-\delta}\right\},$$

where the parameters $\beta \in \mathbb{N}_+$ and $\gamma, \delta \in \mathbb{N}$ can be understood as the basis, the magnitude, and the resolution of the fixed-point numbers, respectively. For example, for $\beta = 10$ and $\gamma = \delta = 1$, we obtain $\mathbb{Q}_{\beta,\gamma,\delta} = \{-10, -9.9, \dots, 9.8, 9.9\}$. Now, different user-defined mappings $h : \mathbb{R} \to \mathbb{Q}_{\beta,\gamma,\delta}$ can be used to compute fixed-point approximations of the form $\hat{\boldsymbol{x}}_j := h(\boldsymbol{x}_j)$. Rounding down can, for example, be realized by

$$h(x) := \max\left(\{-\beta^\gamma\} \cup \{q \in \mathbb{Q}_{\beta,\gamma,\delta} \,|\, q \leq x\}\right).$$

Other rounding schemes can be implemented analogously. It remains to address the mapping from $\mathbb{Q}_{\beta,\gamma,\delta}$ to the message space $\mathbb{N}_P$ of the Paillier cryptosystem. In this context, we first note that

$$\beta^\delta\, \mathbb{Q}_{\beta,\gamma,\delta} = \{-\beta^{\gamma+\delta}, -\beta^{\gamma+\delta}+1, \dots, \beta^{\gamma+\delta}-1\} \subset \mathbb{Z}. \tag{9}$$

Hence, scaling with β^δ maps $\mathbb{Q}_{\beta,\gamma,\delta}$ onto a subset of $\mathbb{Z}$. In order to obtain a mapping onto a subset of $\mathbb{N}_P$, we choose a number $Q \in \mathbb{N}_P$ that is large enough to avoid overflow (as detailed below) and define the function $f : \mathbb{Z} \to \mathbb{N}_Q$ as $f(z) := z \bmod Q$. The combination of the quantization via h, the scaling with b^δ, and the mapping f leads to $f(\beta^\delta h(\boldsymbol{x}_j)) \in \mathbb{N}_Q \subseteq \mathbb{N}_P$. The resulting numbers can then be processed in the Paillier cryptosystem. In this context, it turns out to be useful that the relations

$$f(z_1 + z_2) = f(z_1) + f(z_2) \bmod Q \quad \text{and} \tag{10a}$$

$$f(z_1 z_2) = f(z_1) f(z_2) \bmod Q. \tag{10b}$$

hold for every $z_1, z_2 \in \mathbb{Z}$. Moreover, $\varphi : \mathbb{Z} \to \mathbb{Z}$ with

$$\varphi(z) := \begin{cases} z - Q & \text{if } z \geq \frac{Q}{2}, \\ z & \text{otherwise} \end{cases} \tag{11}$$

is a partial inverse of f. In fact, we have $z = \varphi(f(z))$ for all

$$z \in \left\{-\left\lfloor \frac{Q}{2} \right\rfloor, -\left\lfloor \frac{Q}{2} \right\rfloor + 1, \dots, \left\lceil \frac{Q}{2} \right\rceil - 1\right\}. \tag{12}$$

Invertibility of f is required to undo the mapping onto $\mathbb{N}_P$.

3. NOVEL ENCRYPTED MPC

In this section, we introduce a novel encrypted MPC implementation that builds on an approximate solution of the OCP (2) using ADMM real-time iterations. In order to prepare the novel scheme, we first summarize the ADMM-based solution of (2) in Section 3.1 and recall the corresponding real-time iterations in Section 3.2.

3.1 ADMM-based solution of the OCP

Different strategies have been proposed to solve OCPs of the form (2) using ADMM. Here, we follow the approach from Jerez et al. (2014) that is based on reformulating (2) in terms of the optimization problem

$$\boldsymbol{z}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{z}} \; \frac{1}{2}\, \boldsymbol{z}^\top \boldsymbol{H} \boldsymbol{z} \tag{13a}$$

$$\text{s.t.} \qquad \boldsymbol{z} \in \mathcal{Z}, \tag{13b}$$

$$\boldsymbol{G}\boldsymbol{z} = \boldsymbol{E}\boldsymbol{x} \tag{13c}$$

with the decision variables

$$\boldsymbol{z} := \begin{pmatrix} \tilde{\boldsymbol{u}}(0) \\ \vdots \\ \tilde{\boldsymbol{u}}(N-1) \\ \tilde{\boldsymbol{x}}(1) \\ \vdots \\ \tilde{\boldsymbol{x}}(N) \end{pmatrix} \in \mathbb{R}^p \tag{14}$$

and the constraint set

$$\mathcal{Z} := \left\{\boldsymbol{z} \in \mathbb{R}^p \,\middle|\, \boldsymbol{z} \in \mathcal{U}^N \times \mathcal{X}^{N-1} \times \mathcal{T}\right\}, \tag{15}$$

where $p := N(m+n)$ and where we refer to (Schulze Darup and Book, 2019a, Eq. (2.1)) for details on the matrices $\boldsymbol{H}$, $\boldsymbol{G}$, and $\boldsymbol{E}$. To prepare the application of ADMM, we first rewrite (13) in terms of an unconstrained optimization problem. With the help of indicator functions, we find

$$\boldsymbol{z}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{z}} \; \frac{1}{2}\, \boldsymbol{z}^\top \boldsymbol{H} \boldsymbol{z} + \mathcal{I}_{\mathcal{Z}}(\boldsymbol{z}) + \mathcal{I}_{\mathcal{E}(\boldsymbol{x})}(\boldsymbol{z}),$$

where $\mathcal{E}(\boldsymbol{x}) := \{\boldsymbol{z} \in \mathbb{R}^{Nm+Nn} \,|\, \boldsymbol{G}\boldsymbol{z} = \boldsymbol{E}\boldsymbol{x}\}$. Following the approach in Jerez et al. (2014), we next introduce the "copy" $\boldsymbol{y}$ of $\boldsymbol{z}$ in order to derive the decomposed optimization problem

$$\min_{\boldsymbol{y},\boldsymbol{z}} \; \frac{1}{2}\, \boldsymbol{y}^\top \boldsymbol{H} \boldsymbol{y} + \mathcal{I}_{\mathcal{Z}}(\boldsymbol{z}) + \mathcal{I}_{\mathcal{E}(\boldsymbol{x})}(\boldsymbol{y}) \tag{16a}$$

$$\text{s.t.} \qquad \boldsymbol{y} = \boldsymbol{z}. \tag{16b}$$

From (16), one can derive the three ADMM steps

$$\begin{aligned}\boldsymbol{y}^{(j+1)} := \arg\min_{\boldsymbol{y}}\ & \frac{1}{2}\,\boldsymbol{y}^\top(\boldsymbol{H}+\rho\boldsymbol{I}_p)\boldsymbol{y} + (\boldsymbol{\mu}^{(j)}-\rho\boldsymbol{z}^{(j)})^\top\boldsymbol{y} \\ \text{s.t.}\ & \ \boldsymbol{G}\boldsymbol{y}=\boldsymbol{E}\boldsymbol{x},\end{aligned} \tag{17a}$$

$$\boldsymbol{z}^{(j+1)} := \mathrm{proj}_{\mathcal{Z}}\Big(\boldsymbol{y}^{(j+1)}+\frac{1}{\rho}\boldsymbol{\mu}^{(j)}\Big), \tag{17b}$$

$$\boldsymbol{\mu}^{(j+1)} := \boldsymbol{\mu}^{(j)} + \rho\big(\boldsymbol{y}^{(j+1)}-\boldsymbol{z}^{(j+1)}\big), \tag{17c}$$

where $\rho > 0$ weights the regulariztion term and where $\boldsymbol{\mu} \in \mathbb{R}^p$ represents the Lagrange multipliers (see Jerez et al. (2014) for details). The step (17a) corresponds to an equality-constrained quadratic program. Hence, the optimizer $\boldsymbol{y}^{(j+1)}$ can be inferred from the equation

$$\begin{pmatrix}\boldsymbol{H}+\rho\boldsymbol{I}_p & \boldsymbol{G}^\top \\ \boldsymbol{G} & \boldsymbol{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{y}^{(j+1)} \\ *\end{pmatrix} = \begin{pmatrix}\rho\boldsymbol{z}^{(j)}-\boldsymbol{\mu}^{(j)} \\ \boldsymbol{E}\boldsymbol{x}\end{pmatrix},$$

where "$*$" is irrelevant here. Due to $\boldsymbol{H}$ being positive definite, ρ being positive, and $\boldsymbol{G}$ having full rank, the inverse

$$\boldsymbol{\Gamma} := \begin{pmatrix}\boldsymbol{\Gamma}_{11} & \boldsymbol{\Gamma}_{12} \\ \boldsymbol{\Gamma}_{12}^\top & \boldsymbol{\Gamma}_{22}\end{pmatrix} = \begin{pmatrix}\boldsymbol{H}+\rho\boldsymbol{I}_p & \boldsymbol{G}^\top \\ \boldsymbol{G} & \boldsymbol{0}\end{pmatrix}^{-1}$$

is well-defined and we obtain

$$\boldsymbol{y}^{(j+1)} = \boldsymbol{\Gamma}_{11}\big(\rho\boldsymbol{z}^{(j)}-\boldsymbol{\mu}^{(j)}\big)+\boldsymbol{\Gamma}_{12}\boldsymbol{E}\boldsymbol{x}. \tag{18}$$

Since the resulting steps do not depend on $\boldsymbol{y}^{(j)}$, we can eliminate the copy $\boldsymbol{y}$ by substituting $\boldsymbol{y}^{(j+1)}$ from (18) in (17b) and (17c). The two remaining steps then are

$$\boldsymbol{z}^{(j+1)} = \mathrm{proj}_{\mathcal{Z}}\Big(\boldsymbol{\Gamma}_{11}\big(\rho\boldsymbol{z}^{(j)}-\boldsymbol{\mu}^{(j)}\big)+\boldsymbol{\Gamma}_{12}\boldsymbol{E}\boldsymbol{x}+\frac{1}{\rho}\boldsymbol{\mu}^{(j)}\Big),$$

$$\boldsymbol{\mu}^{(j+1)} = \boldsymbol{\mu}^{(j)}+\rho\big(\boldsymbol{\Gamma}_{11}\big(\rho\boldsymbol{z}^{(j)}-\boldsymbol{\mu}^{(j)}\big)+\boldsymbol{\Gamma}_{12}\boldsymbol{E}\boldsymbol{x}-\boldsymbol{z}^{(j+1)}\big).$$

These iterations are known to converge to the optimum of (13) for any $\rho > 0$. However, an efficient implementation of the iterations requires an efficient computation the projections onto $\mathcal{Z}$. As apparent from (15), a projection onto $\mathcal{Z}$ can be decomposed into a finite number of projections onto $\mathcal{U}$, $\mathcal{X}$, and $\mathcal{T}$ and an efficient computation of these projections is guaranteed by Assumption 2.

3.2 Dynamics for real-time iterations

We usually require multiple ADMM iterations to approximate $\boldsymbol{z}^*(\boldsymbol{x})$ with a certain accuracy. However, as shown in Schulze Darup and Book (2019b), a single iteration per time-step can be sufficient in the framework of MPC. In order to describe the resulting closed-loop behavior, it is convenient to introduce the augmented state

$$\boldsymbol{\xi} := \begin{pmatrix}\boldsymbol{x} \\ \boldsymbol{z}^{(0)} \\ \boldsymbol{\mu}^{(0)}\end{pmatrix} \in \mathbb{R}^{n+2p}$$

and the matrix

$$\boldsymbol{\mathcal{K}} := \begin{pmatrix}\boldsymbol{\Gamma}_{12}\boldsymbol{E} & \rho\boldsymbol{\Gamma}_{11} & \frac{1}{\rho}\boldsymbol{I}_p-\boldsymbol{\Gamma}_{11}\end{pmatrix} \in \mathbb{R}^{p\times(n+2p)}.$$

In fact, based on $\boldsymbol{\xi}$ and $\boldsymbol{\mathcal{K}}$, the previously discussed ADMM iterations can be rewritten in a compact fashion leading to the real-time iterations

$$\boldsymbol{z}^{(1)}(k) = \mathrm{proj}_{\mathcal{Z}}\left(\boldsymbol{\mathcal{K}}\,\boldsymbol{\xi}(k)\right), \tag{19a}$$

$$\boldsymbol{\mu}^{(1)}(k) = \rho\big(\boldsymbol{\mathcal{K}}\boldsymbol{\xi}(k)-\boldsymbol{z}^{(1)}(k)\big). \tag{19b}$$

Now, since only one iteration is evaluated per time step, $\boldsymbol{z}^{(1)}(k)$ is our best guess of $\boldsymbol{z}^*(\boldsymbol{x}(k))$. Hence, we apply the control action

$$\boldsymbol{u}(k) = \boldsymbol{C}\boldsymbol{z}^{(1)}(k) \quad \text{with} \quad \boldsymbol{C} := \begin{pmatrix}\boldsymbol{I}_m & \boldsymbol{0}_{m\times(p-m)}\end{pmatrix}. \tag{20}$$

in analogy to (3) and by definition of $\boldsymbol{z}$ in (14). At time step $k+1$, we will later measure the state $\boldsymbol{x}(k+1)$. The remaining entries of $\boldsymbol{\xi}(k)$, i.e., $\boldsymbol{z}^{(0)}(k+1)$ and $\boldsymbol{\mu}^{(0)}(k+1)$ can, in principle, be freely chosen. It turns out, however, that warmstarts of the form

$$\boldsymbol{z}^{(0)}(k+1) = \boldsymbol{D}_z\boldsymbol{z}^{(1)}(k) \qquad \text{and} \tag{21a}$$

$$\boldsymbol{\mu}^{(0)}(k+1) = \boldsymbol{D}_\mu\boldsymbol{\mu}^{(1)}(k), \tag{21b}$$

that reuse $\boldsymbol{z}^{(1)}(k)$ and $\boldsymbol{\mu}^{(1)}(k)$ from the previous step, are useful. We note that linear updates similar to (21) have also been considered in other real-time iteration schemes such as (Diehl et al., 2005, Sect. 2.2). Moreover, we refer to (Schulze Darup and Book, 2019b, Sect. IV.A) for suitable choices of the matrices $\boldsymbol{D}_z, \boldsymbol{D}_\mu \in \mathbb{R}^{p\times p}$ that, e.g., implement shifts by one time step. In summary, the dynamics of the controlled system can be described by the augmented system

$$\boldsymbol{\xi}(k+1) = \boldsymbol{\mathcal{A}}\boldsymbol{\xi}(k)+\boldsymbol{\mathcal{B}}\,\mathrm{proj}_{\mathcal{Z}}(\boldsymbol{\mathcal{K}}\boldsymbol{\xi}(k)) \tag{22}$$

with the matrices

$$\boldsymbol{\mathcal{A}} := \begin{pmatrix}(\boldsymbol{A} \;\; \boldsymbol{0}_{n\times 2p}) \\ \boldsymbol{0}_{p\times(n+2p)} \\ \rho\boldsymbol{D}_\mu\boldsymbol{\mathcal{K}}\end{pmatrix} \quad \text{and} \quad \boldsymbol{\mathcal{B}} := \begin{pmatrix}\boldsymbol{B}\boldsymbol{C} \\ \boldsymbol{D}_z \\ -\rho\boldsymbol{D}_\mu\end{pmatrix}.$$

Around the augmented origin, the nonlinear dynamics (22) become linear. It can be shown that the corresponding dynamics, i.e., $\boldsymbol{\xi}(k+1) = (\boldsymbol{\mathcal{A}}+\boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{K}})\boldsymbol{\xi}(k)$, are asymtotically stable for a suitable choice of ρ (Schulze Darup and Book, 2019a, Prop. 6). However, attraction does not only hold under the linear regime. In fact, as apparent from the numerical benchmark in (Schulze Darup and Book, 2019b, Sect. IV), the domain of attraction usually also includes regions with nonlinear dynamics.

3.3 Encrypted implementation in the cloud

We are now ready to derive the encrypted implementation of the proposed controller. To this end, we first note that the control scheme consists of the real-time iterations (19), the control actions (20), and the warmstarts (21). Apparently, all involved operations are linear in the augmented states except of the projection in (19a) that is reused in (19b). As already mentioned in the introduction, an encrypted evaluation of projections is difficult. Hence, we follow the approach in Schulze Darup et al. (2018a) and outsource projections from the cloud to the actuator. In the cloud, we will thus compute

$$\boldsymbol{\zeta}(k) := \boldsymbol{\mathcal{K}}\boldsymbol{\xi}(k) \tag{23}$$

in an encrypted fashion (that is detailed below) and send the result to the actuator, where we decrypt it, evaluate $\boldsymbol{z}^{(1)}(k) = \mathrm{proj}_{\mathcal{Z}}(\boldsymbol{\zeta}(k))$, and apply (20). According to (21a), $\boldsymbol{z}^{(1)}(k)$ is also required to warmstart $\boldsymbol{z}^{(0)}(k+1)$ and, consequently, to define $\boldsymbol{\xi}(k+1)$. In other words, the cloud needs access to $\boldsymbol{z}^{(1)}(k)$ to compute $\boldsymbol{\zeta}(k+1)$ at the next time step. Additionally, $\boldsymbol{x}(k+1)$ and $\boldsymbol{\mu}^{(0)}(k+1)$ are required. However, as apparent from the following proposition, $\boldsymbol{\mu}^{(0)}(k+1)$ can be inferred from $\boldsymbol{z}^{(1)}(k)$ and $\boldsymbol{\zeta}(k)$. Hence, there is no need to explicitly evaluate or communicate $\boldsymbol{\mu}^{(1)}(k)$.

Proposition 3. The augmented state at time step $k+1$ can be inferred from the relation

$$\boldsymbol{\xi}(k+1) = \boldsymbol{\mathcal{L}}\begin{pmatrix}\boldsymbol{x}(k+1) \\ \boldsymbol{z}^{(1)}(k) \\ \boldsymbol{\zeta}(k)\end{pmatrix}, \tag{24}$$

where

$$\boldsymbol{\mathcal{L}} := \begin{pmatrix}\boldsymbol{I}_n & \boldsymbol{0}_{n\times p} & \boldsymbol{0}_{n\times p} \\ \boldsymbol{0}_{p\times n} & \boldsymbol{D}_z & \boldsymbol{0}_{p\times p} \\ \boldsymbol{0}_{p\times n} & -\rho\boldsymbol{D}_\mu & \rho\boldsymbol{D}_\mu\end{pmatrix}.$$

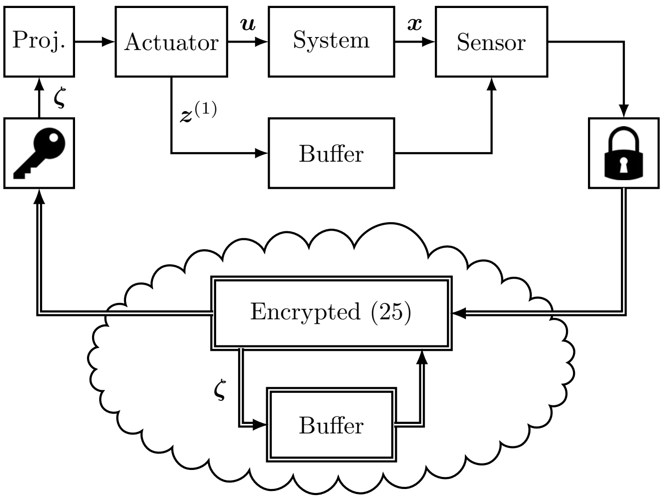

Fig. 2. Encrpyted MPC via ADMM real-time iterations. Buffers are used to provide $\boldsymbol{z}^{(1)}(k-1)$ and $\boldsymbol{\zeta}(k-1)$.

Proof. We have

$$\boldsymbol{\xi}(k+1) = \begin{pmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{z}^{(0)}(k+1) \\ \boldsymbol{\mu}^{(0)}(k+1) \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{D}_z \boldsymbol{z}^{(1)}(k) \\ \boldsymbol{D}_\mu \boldsymbol{\mu}^{(1)}(k) \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{D}_z \boldsymbol{z}^{(1)}(k) \\ \rho \boldsymbol{D}_\mu \left(\boldsymbol{\mathcal{K}}\boldsymbol{\xi}(k) - \boldsymbol{z}^{(1)}(k) \right) \end{pmatrix}$$

by definition of $\boldsymbol{\xi}$, due to the warmstart (21), and according to (19b), respectively. Taking (23) and the definition of $\boldsymbol{\mathcal{L}}$ into account, it becomes apparent that the latter expression is equivalent to the right-hand side in (24). ■

Aiming for the actual controller implementation, it is slightly more intuitive to consider the current step k and its precursor $k-1$ instead of $k+1$ and k as in (24). The central computation of $\boldsymbol{\zeta}(k)$ then results from

$$\boldsymbol{\zeta}(k) = \boldsymbol{\mathcal{M}} \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{z}^{(1)}(k-1) \\ \boldsymbol{\zeta}(k-1) \end{pmatrix}, \tag{25}$$

where $\boldsymbol{\mathcal{M}} := \boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{L}}$. Relation (25) is applied for every time step $k > 0$. More precisely, it is assumed that the sensor provides the current state $\boldsymbol{x}(k)$ and the previous iterate $\boldsymbol{z}^{(1)}(k-1)$ and that the cloud stored $\boldsymbol{\zeta}(k-1)$ in the previous step. The cloud then computes $\boldsymbol{\zeta}(k)$ and submits it to the actuator. As previously discussed, the actuator computes $\boldsymbol{z}^{(1)}(k)$ via projection, applies (20), and forwards $\boldsymbol{z}^{(1)}(k)$ to the sensor. The resulting control loop is illustrated in Figure 2. At the initial step $k = 0$, the original relation (23) is utilized with $\boldsymbol{\mu}^{(0)}(0) = \boldsymbol{0}_p$. Moreover, it is assumed that the sensor provides $\boldsymbol{x}(0)$ and $\boldsymbol{z}^{(0)}(0)$ instead of $\boldsymbol{z}^{(1)}(-1)$. The cloud then computes

$$\boldsymbol{\zeta}(0) = \boldsymbol{\mathcal{K}}\boldsymbol{\xi}(0) = \boldsymbol{\Gamma}_{12}\boldsymbol{E}\boldsymbol{x}(0) + \rho\boldsymbol{\Gamma}_{11}\boldsymbol{z}^{(0)}(0).$$

It remains to comment on the encrypted implementation of the proposed controller. For brevity, we focus on the steps $k > 0$, i.e., the encrypted evaluation of (25). As discussed in Section 2.3, the application of the Paillier cryptosystem requires quantiziations of the involved controller parameters and (augmented) states. To this end, we introduce the quantizations $\widehat{\boldsymbol{\mathcal{M}}}_{ij} := h(\boldsymbol{\mathcal{M}}_{ij})$, $\hat{\boldsymbol{x}}_j(k) := h\big(\boldsymbol{x}_j(k)\big)$, and $\hat{\boldsymbol{z}}_j^{(1)}(k-1) := h\big(\boldsymbol{z}_j^{(1)}(k-1)\big)$ and define

$$\check{\boldsymbol{\zeta}}_i(k) := \sum_{j=1}^{n} \widehat{\boldsymbol{\mathcal{M}}}_{ij}\hat{\boldsymbol{x}}_j(k) + \sum_{j=n+1}^{n+p} \widehat{\boldsymbol{\mathcal{M}}}_{ij}\hat{\boldsymbol{z}}^{(1)}_{j-n}(k-1) + \sum_{j=n+p+1}^{n+2p} \widehat{\boldsymbol{\mathcal{M}}}_{ij}\check{\boldsymbol{\zeta}}_{j-n-p}(k-1) \tag{26}$$

for every $i \in \{1, \dots, p\}$. In this context, it is important to note that $\widehat{\boldsymbol{\mathcal{M}}}_{ij}, \hat{\boldsymbol{x}}_j(k), \hat{\boldsymbol{z}}_j^{(1)}(k-1) \in \mathbb{Q}_{\beta,\gamma,\delta}$ holds by construction whereas $\check{\boldsymbol{\zeta}}_i(k)$ and $\check{\boldsymbol{\zeta}}_i(k-1)$ may or may not be contained in $\mathbb{Q}_{\beta,\gamma,\delta}$. We study products of the form $\widehat{\boldsymbol{\mathcal{M}}}_{ij}\hat{\boldsymbol{x}}_j(k)$ to clarify this observation. Apparently, multiplying two numbers from a quantization grid with a resolution of $\beta^{-\delta}$, e.g., from $\mathbb{Q}_{\beta,\gamma,\delta}$, yields a product that may refer to a quantization with a resolution of $\beta^{-2\delta}$. For numerical example, consider 0.1 and 0.5 from $\mathbb{Q}_{10,1,1}$ and note that $0.1 \times 0.5 = 0.05 = 5 \times 10^{-2}$. Now, let us assume $\check{\boldsymbol{\zeta}}_i(k)$ requires a resolution of $\beta^{-2\delta}$, then the reutilization of $\check{\boldsymbol{\zeta}}_i(k)$ in the subsequent time step may result in a resolution of $\beta^{-3\delta}$ for $\check{\boldsymbol{\zeta}}_i(k+1)$. Hence, for $k \to \infty$, the required resolution will be infinitely small. This is impractical since the resolution influences the mapping of the quantized data onto the integer message space of Paillier (cf. (9)). In order to avoid this drawback, we refresh the quantization of $\check{\boldsymbol{\zeta}}_i(k)$ every $\Delta k \in \mathbb{N}_+$ time steps. The role of Δk will be specified in the following.

As illustrated in Figure 2, the encrypted evaluation of (26) starts at the sensor with the encryption of $\hat{\boldsymbol{x}}(k)$ and $\hat{\boldsymbol{z}}^{(1)}(k-1)$. More precisely, at time steps $k > 0$ with $k \bmod \Delta k \neq 0$, we compute the ciphertexts

$$\boldsymbol{c}_j(k) := \texttt{Enc}\left(f\big(\beta^{(1+(k \bmod \Delta k))\delta}\hat{\boldsymbol{x}}_j(k)\big), \boldsymbol{r}_j(k) \right) \tag{27a}$$

for $j \in \{1, \dots, n\}$ and

$$\boldsymbol{c}_j(k) := \texttt{Enc}\Big(f\big(\beta^{(1+(k \bmod \Delta k))\delta}\hat{\boldsymbol{z}}^{(1)}_{j-n}(k-1)\big), \boldsymbol{r}_j(k) \Big) \tag{27b}$$

for $j \in \{n+1, \dots, n+p\}$ according to (5), where the numbers $\boldsymbol{r}_j(k)$ are randomly chosen from $\mathbb{Z}_P^*$. The encrypted vector $\boldsymbol{c}(k)$ is then send to the cloud, where the ciphertexts

$$\boldsymbol{v}_i(k) := \left(\prod_{j=1}^{n+p} \boldsymbol{c}_j(k)^{\boldsymbol{Z}_{ij}} \right) \left(\prod_{j=n+p+1}^{n+2p} \boldsymbol{v}_{j-n-p}(k-1)^{\boldsymbol{Z}_{ij}} \right) \bmod P^2$$

are evaluated and stored for every $i \in \{1, \dots, p\}$ with $\boldsymbol{Z}_{ij} := f\big(\beta^\delta \widehat{\boldsymbol{\mathcal{M}}}_{ij}\big)$. Next, the encrypted vector $\boldsymbol{v}_i(k)$ is transmitted to the actuator, where $\check{\boldsymbol{\zeta}}(k)$ is recovered from

$$\check{\boldsymbol{\zeta}}_i(k) = \beta^{-(2+(k \bmod \Delta k))\delta} \varphi\left(\texttt{Dec}\big(\boldsymbol{v}_i(k)\big) \bmod Q \right) \tag{28}$$

using (6) and (11), where we note that a correct inversion via φ requires that the argument of the function is contained in (12). Under the assumption that correctness held for step $k-1$, it is straightforward to verify correctness of the encrypted scheme for $k = 1$, i.e., equivalence of $\check{\boldsymbol{\zeta}}_i(k)$ in (26) and (28). In fact, a formal proof can be derived analogously to (Schulze Darup et al., 2018a, Proof of Thm. 1) using the relations (7), (8), and (10). However, further investigating the role of Δk is more elucidating.

The special treatment of steps $k > 0$ with $k \bmod \Delta k = 0$ starts at the actuator already during the preceding step. In fact, if $k+1 \bmod \Delta k = 0$ is recognized at the actuator, then $\check{\boldsymbol{\zeta}}(k)$ is forwarded to the sensor alongside with

$\boldsymbol{z}^{(1)}(k) = \mathrm{proj}_{\mathcal{Z}}(\check{\boldsymbol{\zeta}}(k))$. Now, at the following time step, the sensor measures $\boldsymbol{x}(k)$ and collects the buffered $\boldsymbol{z}^{(1)}(k-1)$ and $\check{\boldsymbol{\zeta}}(k-1)$. It quantizes all three vectors and handles $\hat{\boldsymbol{x}}(k)$ and $\hat{\boldsymbol{z}}^{(1)}(k-1)$ as usual. In addition, it encrypts $\hat{\boldsymbol{\zeta}}_j(k-1) := h\left(\check{\boldsymbol{\zeta}}_j(k-1)\right) \in \mathbb{Q}_{\beta,\gamma,\delta}$ by evaluating

$$\boldsymbol{v}_{j-n-p}(k-1) := \mathtt{Enc}\Big(f\big(\beta^{\delta}\hat{\boldsymbol{\zeta}}_j(k-1)\big), \boldsymbol{r}_j(k)\Big)$$

for every $j \in \{n+p+1, \ldots, n+2p\}$ using random numbers $\boldsymbol{r}_j(k) \in \mathbb{Z}_P^*$. Next, the sensor sends $\boldsymbol{v}(k-1)$ to the cloud alongside with $\boldsymbol{c}(k)$. Finally, the cloud overwrites its internal buffer with the ciphertexts $\boldsymbol{v}(k-1)$ and proceeds as usual.

Apparently, Δk affects the encryption in (27) and the decryption in (28). The leading idea is to "count" the number of steps since the last refreshment and to adapt the mapping onto and the recovery from the integer message space accordingly. The adaption is necessary since the required resolution for the representation of $\check{\boldsymbol{\zeta}}(k)$ is changing with k due to the recursive nature of (26).

It remains to comment on the numerical complexity of the proposed scheme. In this context, the numbers of elementary operations per component are itemized in Table 1. In the cloud, modular exponentiations (`Exp`) and modular multiplications (`Mul`) are fundamental for the evaluation of $\boldsymbol{v}_i(k)$ (as specified above). During the refreshment of $\check{\boldsymbol{\zeta}}(k-1)$ or, more precisely $\boldsymbol{v}_i(k-1)$, p additional encryptions are required at the sensor.

Table 1. Number of operations performed at the sensor, in the cloud, and at the actuator.

Mode	Sensor	Cloud		Actuator
	`Enc`	`Exp`	`Mul`	`Dec`
normal	$n+p$	$p(n+2p)$	$p(n+2p-1)$	p
refresh	$n+2p$	$p(n+2p)$	$p(n+2p-1)$	p

4. NUMERICAL EXAMPLE

We apply the proposed scheme to the double-integrator system with the matrices

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{B} := \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$$

and the constraints $\mathcal{X} := \{\boldsymbol{x} \in \mathbb{R}^2 \,|\, |\boldsymbol{x}_1| \leq 25, |\boldsymbol{x}_2| \leq 5\}$ and $\mathcal{U} := [-1, 1]$. We consider the weighting matrices $\boldsymbol{Q} = \boldsymbol{I}_2$ and $\boldsymbol{R} = 0.1$. The prediction horizon is set to $N = 12$ and the terminal weighting $\boldsymbol{P}$ is chosen as the solution of the (discrete-time) algebraic Riccati equation

$$\boldsymbol{A}^\top(\boldsymbol{P} - \boldsymbol{P}\,\boldsymbol{B}\,(\boldsymbol{R} + \boldsymbol{B}^\top\boldsymbol{P}\,\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{P})\,\boldsymbol{A} - \boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{0}_{n\times n}.$$

We set $\mathcal{T} = \mathcal{X}$ for simplicity. We note, however, that an ellipsoidal terminal set of the form

$$\mathcal{T} := \left\{\boldsymbol{x} \in \mathbb{R}^2 \,|\, \|\boldsymbol{x}\|_{\boldsymbol{P}}^2 \leq c\right\}$$

could be used to enforce stability. In fact, ellipsoidal sets allow for efficient projections as required by Assumption 2. In contrast, polytopic sets $\mathcal{T}$ that are frequently used in MPC do, in general, not support efficient projections.

The ADMM real-time iterations are parametrized as follows. We select the weighting factor $\rho = 10$. The initial guess for $\boldsymbol{z}^{(0)}$ is chosen as

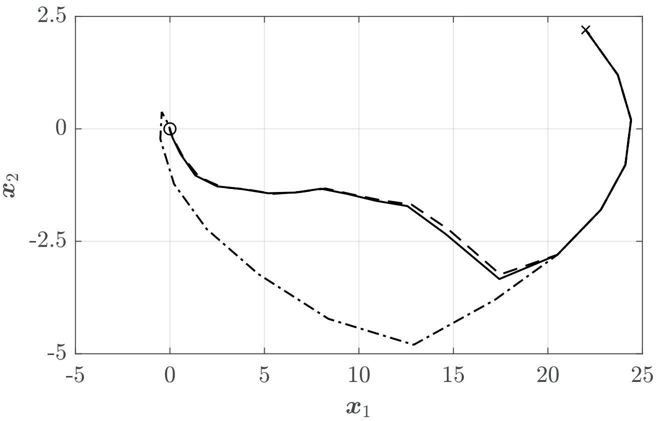

Fig. 3. Illustration of state trajectories resulting for the quantized and encrypted ADMM real-time iterations scheme (solid), for the precise scheme (dashed), and for the original MPC (dash-dotted).

$$\boldsymbol{z}^{(0)}(0) := \begin{pmatrix} \boldsymbol{K}^*(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}^*)^0 \\ \vdots \\ \boldsymbol{K}^*(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}^*)^{N-1} \\ (\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}^*)^1 \\ \vdots \\ (\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}^*)^N \end{pmatrix} \boldsymbol{x}_0,$$

where $\boldsymbol{K}^* := (\boldsymbol{R} + \boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{A}$ reflects the linear-quadratic regulator (LQR). The updates matrices $\boldsymbol{D}_z$ and $\boldsymbol{D}_\mu$ are chosen as in Schulze Darup and Book (2019b) for the case "shift-LQR". Hence, the warmstarts (21) implement a shift of the current sequences by one time step followed by an LQR step to complete $\boldsymbol{z}^{(0)}(k+1)$.

It remains to specify the quantization and encryption. In this context, we select $\beta = 2$, $\gamma = 5$, and $\delta = 10$. Hence, we consider signed 16-bit quantizations with 5 integer bits and 10 fractional bits. Moreover, we choose $Q = 2^{128}$ (to ensure correct inversions via φ) and prime numbers p_1 and p_2 of length $l = 1024$ (as suggested by the National Institute of Standards and Technology (2016)). Finally, we refresh the quantization of $\check{\boldsymbol{\zeta}}$ every $\Delta k = 5$ time steps.

The closed-loop dynamics for the initial state $\boldsymbol{x}_0 := (22 \quad 2.2)^\top$ are illustrated in Figure 3. More precisely, the state trajectory for the proposed scheme including quantization and encryption is shown in comparison to the results for ADMM real-time iterations without quantization and the original MPC. As apparent from the figure, all variants steer the system towards the origin. Moreover, the quantized ADMM scheme shows only small deviations from the precise one. Compared to the original MPC, we observe an identical behavior for the first five time steps. Afterwards, significant differences between the real-time iterations and the optimal solution are visible.

For completeness, we briefly comment on the numerical effort to run the example. As apparent from Table 1, the number of required operations is determined by $n = 2$ and $p = N(m+n) = 36$. Hence, 38 encryptions, 2664 modular exponentations, 2628 modular multiplications, and 36 decryptions are required during "normal" time steps. During the refreshment step, the number of encryptions increases to 74. Taking into account that these operations are numerically demanding, it is not surprising that a simple Python implementation (without parallelization) running on a 2.7 GHz Intel Core i7-7500U requires 5.7 s at the sensor (11.2 s for refreshment step), 12.5 s in the cloud, and 5.2 s at the actuator.

5. CONCLUSION AND OUTLOOK

We presented a novel encrypted MPC that is based on an encrypted implementation of ADMM real-time iterations. Compared to the existing schemes Alexandru et al. (2018) and Schulze Darup et al. (2018a), the novel encrypted controller stands out for the ability to consider state and input constraints. Furthermore, in contrast to Schulze Darup et al. (2018b), an explicit solution of the underlying optimal control problem (OCP) is not required.

From a technical point of view, the novel scheme is interesting since the controller can be formulated in a recursive manner (see Prop. 3), which reduces the communication and decryption load. However, recursively handling encrypted data in the cloud requires, in general, to refresh the quantization from time to time. A suitable strategy has been derived and successfully implemented. We note, at his point, that the computations (25) carried out in the cloud can be interpreted as a linear dynamical controller. Hence, instead of regular refreshments of the quantized controller state $\boldsymbol{\zeta}$, tailored approaches as, e.g., proposed in Cheon et al. (2018) could be an interesting alternative. Moreover, realizing encrypted projections would further promote the role of the cloud and relieve the actuator.

Finally, from a control perspective, the numerical example illustrates the functionality of the scheme. However, the differences between the three trajectories in Figure 3 suggest some directions for further research. In fact, the effect of quantization errors has not yet been addressed formally. Further, apart from encrypted control, a closer investigation of the performance of ADMM real-time iterations compared to the optimal solutions is desirable. Lastly, we aim for a more efficient implementation of the proposed scheme in order to reduce the required sampling time.

REFERENCES

Alexandru, A.B., Morari, M., and Pappas, G.J. (2018). Cloud-based MPC with encrypted data. In *Proc. of the 57th Conference on Decision and Control*, 5014–5019.

Alexandru, A.B. and Pappas, G.J. (2019). Encrypted LQG using labeled homomorphic encryption. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 129–140.

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2014). (Leveled) Fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3), 13:1–13:36.

Cheon, J.H., Han, K., Kim, H., Kim, J., and Shim, H. (2018). Need for controllers having integer coefficients in homomorphically encrypted dynamic system. In *Proc. of 57th Conference on Decision and Control*, 5020–5025.

Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12, 577–585.

Diehl, M., Findeisen, R., Allgöwer, F., Bock, H.G., and Schlöder, J.P. (2005). Nominal stability of real-time iteration scheme for nonlinear model predictive control. *IEE Proceedings - Control Theory and Applications*, 152(3), 296–308.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4), 469–472.

Farokhi, F., Shames, I., and Batterham, N. (2017). Secure and private control using semi-homomorphic encryption. *Control Engineering Practice*, 67, 13–20.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 169–178.

Gentry, C. (2010). Computing arbitrary functions of encrypted data. *Communications of the ACM*, 22(11), 612–613.

Jerez, J.L., Goulart, P.J., Richter, S., Constantinides, G.A., Kerrigan, E.C., and Morari, M. (2014). Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12), 3238–3251.

Katz, J. and Lindell, Y. (2014). *Introduction to Modern Cryptography, Second Edition*. CRC Press.

Kim, J., Lee, C., Shim, H., Cheon, J.H., Kim, A., Kim, M., and Song, Y. (2016). Encrypting controller using fully homomorphic encryption for security of cyber-physical systems. In *Proc. of the 6th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 175–180.

Kogiso, K. and Fujita, T. (2015). Cyber-security enhancement of networked control systems using homomorphic encryption. In *Proc. of the 54th Conference on Decision and Control*, 6836–6843.

Li, W.C. and Biegler, L.T. (1989). A multistep, newton-type control strategy for constrained, nonlinear processes. In *Proc. of the 1989 American Control Conference*, 1526–1527.

National Institute of Standards and Technology (2016). Recommendation for key management. In E. Barker (ed.), *NIST Special Publication 800-57 Part 1*.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, 223–238. Springer.

Schulze Darup, M. and Book, G. (2019a). On closed-loop dynamics of ADMM-based MPC. arXiv:1911.02641 [math.OC].

Schulze Darup, M. and Book, G. (2019b). Towards real-time ADMM for linear MPC. In *Proc. of the 2019 European Control Conference*, 4276–4282.

Schulze Darup, M., Redder, A., and Quevedo, D.E. (2018a). Encrypted cloud-based MPC for linear systems with input constraints. In *Proc. of 6th IFAC Nonlinear Model Predictive Control Conference*, 635–642.

Schulze Darup, M., Redder, A., Shames, I., Farokhi, F., and Quevedo, D. (2018b). Towards encrypted MPC for linear constrained systems. *IEEE Control Systems Letters*, 2(2), 195–200.

van Parys, R. and Pipeleers, G. (2018). Real-time proximal gradient method for linear MPC. In *Proc. of the 2018 European Control Conference*, 1142–1147.