2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Approximated Explicit NMPC via Reinforcement Learning for Homomorphically Encrypted Process Control

Diana Dzurková, Patrik Valábek, Olivér Mészáros, Martin Kalúz, Martin Klaučo
Institute of Information Engineering, Automation, and Mathematics
Slovak University of Technology in Bratislava
Bratislava, Slovakia
diana.dzurkova@stuba.sk

***Abstract*—This research proposes a novel approach to generating explicit, nearly-optimal (suboptimal) control policies in the form of neural networks with a structure that allows further mathematical operations within homomorphic encryption frameworks. The novelty of this paper also lies in presenting a reinforcement learning pathway to train the explicit control law without the necessity of prior model knowledge. A Deep Deterministic Policy Gradient algorithm is used to train the neural network, with the objective function adopted from nonlinear model predictive control. This paper presents a generalized methodology to train the control policy and evaluate it in a homomorphic encryption setup. Particular results are presented based on a software-in-the-loop simulation setup, where specifics like communication delays and computational overheads are considered.**

***Index Terms*—secure process control, data-driven explicit control law, reinforcement learning, homomorphic encryption, data privacy**

I. Introduction

Data privacy plays a vital role in every aspect of the implementation of process control applications. In recent history, several studies have documented tragic breaches in security that led to service disruption, damaged processes, and decreased productivity [1], [2]. Even though data security is primarily the domain of computer science and IT specialists [3], the integration of encryption in the process control layer can greatly improve the overall security of individual process control applications.

The use of homomorphic encryption in the process control domain can ensure security on several levels. Firstly, it allows the encryption of measurement data sent to the controller for the evaluation of control actions. Secondly, it even allows for performing mathematical operations with an encrypted controller, an ability that sets it apart from traditional encryption standards like RSA or AES which require decryption before any operations over the data. Such concepts directly ensure not only data privacy but also the security of the control algorithm itself [4].

Recent advancements in homomorphic encryption (HE) and its application in process control demonstrate how to employ several control algorithms, such as linear state feedback [5], polynomial controllers [6], and PID controllers [7], to govern processes securely. So far, the application of HE frameworks has been limited to explicit control laws [8], making the traditional application of optimization-based strategies unfeasible. This is due to the limited operations supported on homomorphically encrypted data (primarily addition and multiplication), the absence of comparison functions, and the high computational complexity and memory requirements. However, the use of neural networks (NN) as controller approximations can overcome this limitation. The NN controller can be trained to approximate the control law, and the trained NN can be evaluated over encrypted data, yielding encrypted control actions [6].

This paper explores the unique combination of generating approximate explicit control laws with reinforcement learning approaches, closely resembling the performance of the nonlinear model predictive strategy. We employ the Deep Deterministic Policy Gradient (DDPG) algorithm [9] to train the HE-friendly explicit control law. The contributions of this paper are twofold: (1) we demonstrate the application of the DDPG algorithm to train an explicit control law, and (2) we present the integration of the trained control law with homomorphic encryption to ensure data privacy in the process domain.

The structure of the paper includes the problem statement in Section II, followed by a methodology section devoted to cryptosystems and reinforcement learning (Sections III and IV, respectively). A benchmark system and case study are presented in Section V, while the deployment of the encrypted

The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants VEGA 1/0490/23, the Slovak Research and Development Agency under the project APVV-21-0019 and APVV-20-0261. This paper is also funded by the European Union's Horizon Europe under grant no. 101079342 (Fostering Opportunities Towards Slovak Excellence in Advanced Control for Smart Industries). Patrik Valábek and Diana Dzurková thank for financial contribution from the STU Grant Scheme for Support of Young Researchers.

process control, including runtimes and implementation details, is presented in Section VI.

II. Problem Statement

This paper proposes an approach that enforces data privacy associated with a cloud-based evaluation of a nearly optimal control policy. More advanced control methods like MPC are slowly gaining traction across industries, which is directly linked to an increase in computational complexity and its potential outsourcing using cloud-based computing. Data privacy is guaranteed by calculating control action in homomorphic setup, hence the cloud evaluator has no knowledge of the data. We aim to design a control policy that has the following properties:

- is nearly-optimal, and the policy adheres to the constraints of the system,
- the policy formulated as explicit control law, hence no iterative algorithm is required for its evaluation.

To achieve these properties, we propose to use the Deep Deterministic Policy Gradient (DDPG) algorithm, which is a model-free reinforcement learning algorithm that can be used to train a neural network (NN) to generate an explicit control law based on objective function adopted from nonlinear model predictive control (NMPC). Moreover, individual nodes of the NN are of the polynomial form, which allows us to evaluate the control action in a homomorphic setup. The contributions of this work can be summarized as follows:

1) We have formulated an optimization-based control strategy that is inherently compatible with HE. This design enables the execution of control policies in environments where data privacy is paramount, without significant effects to the control's effectiveness or efficiency.
2) Our methodology extends to the practical deployment of encrypted optimal control within a Software-in-the-Loop (SIL) framework. This application demonstrates the feasibility of executing approximated optimal control policies under encryption, a crucial step toward real-world implementation. By validating our approach in a SIL setup, we are stepping beyond the traditional simulation-based evaluations commonly found in the literature. [10], [11]
3) We also provide benchmarks connected to this deployment. Communications are facilitated over HTTPS within a local network where two virtual machines (VM) communicate to mimic real-world conditions more closely. This benchmark focuses on encrypted closed-loop control, capturing performance metrics like operational time and memory usage across both the key exchange phase and the control loop runtime. Our findings offer novel insights into the practical deployment challenges and performance of homomorphically encrypted control systems, marking a significant step towards bridging the gap between theoretical constructs and their practical applicability in safeguarding data privacy within cloud-based control systems.

Overall, this paper contributes significantly to the advancement of privacy-preserving control systems, offering a novel approach that combines the rigor of optimal control policies with the demands of data privacy. Through our innovative design, practical deployment strategies, and insightful benchmarks, we pave the way for future applications in this critical intersection of control theory and data privacy.

III. CKKS Cryptosystem

We have opted to integrate the Cheon-Kim-Kim-Song (CKKS) cryptosystem into our application due to its inherent capability to handle operations over real numbers, thereby obviating the need for supplementary encoding on the user's end. Furthermore, CKKS offers advantageous functionalities such as vector rotation and batching, enabling computations across multiple slots within a ciphertext, thereby facilitating evaluation of multiple data in a single operation. This characteristic proves particularly advantageous for applications based on neural networks.

Since the CKKS belongs to "leveled" cryptosystems, the multiplicative depth of an arithmetic circuit is one of the most limiting computational factors. Each ciphertext has a certain amount of cryptographic noise (as well as approximation error for CKKS schemes), which increases as more operations are performed on it. In case of this noise exceeds a certain level, referred to as the "noise budget", it would ultimately result in the corruption of the original message hidden inside a ciphertext. To prevent the error from growing exponentially while multiplying the ciphertext, CKKS uses a modulus-switching method known as rescaling. However, this mechanism can be performed only a limited number of times, which is usually referred to as the multiplicative depth of the cryptosystem. More information about CKKS and how to set it up can be found in [5], [12].

IV. Deep Deterministic Policy Gradient

Deep deterministic policy gradient (DDPG) is a model-free reinforcement learning algorithm based on the actor-critic method [13]. The DDPG approach trains a neural network, that serves as the substitute for the controller. By only interacting with the model of the process, it learns a control policy, that has similar features as the one obtained by the model predictive control (MPC) algorithm [14]. To achieve this similarity, we set up the reward function in a way that resembles the cost function used in MPC. Since, the outcome of the training is a neural network, the control policy is deterministic and is in the form of explicit function.

In the actor-critic framework, the actor denoted f_{ACT} represents the control policy, i.e., the control law. The critic denoted f_{CRIT} is an approximation of the value function [15, Sec. 3.5]. Both actor and critic components are cast as neural networks of user-defined structure. This structure directly dictates cryptosystem set up, more importantly the its multiplicative depth(more in Section V-C). Based on the f_{CRIT}, the weights of both f_{ACT} and f_{CRIT} are updated during the offline training phase. After sufficient training, we switch to

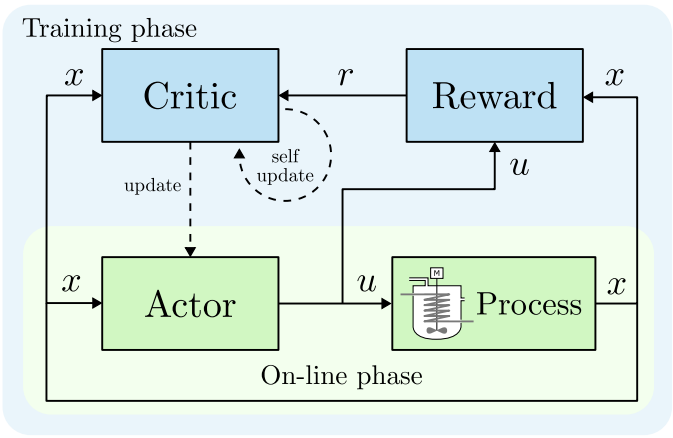

Fig. 1: Schematic overview of DDPG algorithm. The dashed lines indicate that based on the critic, both actor and critic weights are updated during the offline phase. After sufficient training, we switch to the online phase, where the actor is no longer updated and works as a deterministic controller.

the online phase, where f_{ACT} is no longer updated and serves as the deterministic controller. The schematic overview of the DDPG algorithm is shown in Figure 1.

The objective of the DDPG is to find the optimal control policy $u_{\text{ACT}} = f_{\text{ACT}}(x)$ that minimizes a cumulative value of the cost function. The cost function is defined as:

$$J_{\text{DDPG}}(x,u) = \left(x_1^\intercal Q_{\text{x}} x_1 + u_0^\intercal Q_{\text{u}} u_0\right), \tag{1}$$

where x_1 represents the induced state vector by applying u_0 control vector on the previous state x_0. The matrices Q_{x} and Q_{u} are the weighting matrices for the state and control variables, respectively, and their values are determined according to (4). The algorithm aims to find the f_{ACT} that minimizes the cumulative value of the objective function $J_{\text{DDPG}}(x,u)$ over a predefined simulation length. The constraints can be applied to the DDPG algorithm by using a barrier function to penalize the violation of the constraints in the cost function. In the presented case study, the input constraints are inherently enforced by the clipping nature of the output from the algorithm. Conversely, the output of the DDPG framework is a generated explicit control law with nearly optimal performance. Detailed comparison will be presented in the following sections.

V. Benchmark Simulation Model

A. Continuous Stirred-Tank Reactor

First-order irreversible exothermic reaction $A \longrightarrow B$ is taking place inside the CSTR (Fig. 2), where B is the main product. The reaction mixture with initial concentrations $c_{\text{A0}}, c_{\text{B0}}$ and temperature T_{r0} is entering the reactor vessel at the flow rate q_{r}. After the reaction takes place, the product mixture, with final concentrations $c_{\text{A}}, c_{\text{B}}$ and temperature T_{r}, exits the vessel at the same flow rate q_{r}. The heat produced by the reaction is taken away by a coolant with flow rate q_{c}, inlet temperature T_{c0}, and outlet temperature T_{c}. The control task is to retain the reactor in a selected operating

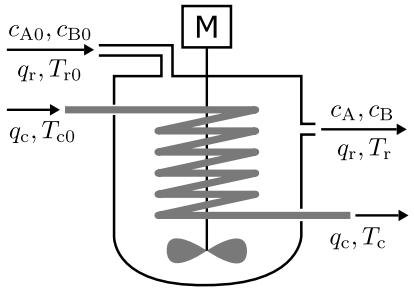

Fig. 2: Schematic diagram of CSTR

TABLE I: Parameters of CSTR model

Symbol	Meaning	Value/Unit
$T_{\text{r0}}, T_{\text{c0}}$	inlet temperatures of reaction mixture and coolant	$325\,\text{K}$, $288\,\text{K}$
$c_{\text{A0}}, c_{\text{B0}}$	inlet concentrations of compounds A and B	$4.22\,\text{mol}\,\text{m}^{-3}$, $0\,\text{mol}\,\text{m}^{-3}$
$\rho_{\text{r}}, \rho_{\text{c}}$	density of reaction mixture and coolant	$1020\,\text{kg}\,\text{m}^{-3}$, $998\,\text{kg}\,\text{m}^{-3}$
$C_{\text{r}}, C_{\text{c}}$	specific heat capacities	$4.02\,\text{J}\,\text{kg}^{-1}\,\text{K}^{-1}$, $4.18\,\text{J}\,\text{kg}^{-1}\,\text{K}^{-1}$
$\nu_{\text{A}}, \nu_{\text{B}}$	stoichiometric coefficients	$1, 1$
h	heat transfer coefficient	$42.8\,\text{W}\,\text{m}^{-2}\,\text{K}^{-1}$
$V_{\text{r}}, V_{\text{c}}$	volume of reactor and cooling system	$0.23\,\text{m}^3$, $0.21\,\text{m}^3$
A	heat transfer area	$1.51\,\text{m}^2$
K_{r}	reaction frequency factor	$1.55 \times 10^{11}\,\text{s}^{-1}$
E_{r}	reaction activation energy	$8.1 \times 10^4\,\text{J}\,\text{mol}^{-1}$
H_{r}	reaction enthalpy	$-6.4 \times 10^4\,\text{J}\,\text{mol}^{-1}$
κ	reaction rate	$f(T_{\text{r}})$ $[\text{s}^{-1}]$

point represented by a steady state. The selected steady state inputs are $u^{\text{s}} = [q_{\text{r}}^{\text{s}}, q_{\text{c}}^{\text{s}}]^\intercal = [0.015, 0.004]^\intercal$ and states are $x^{\text{s}} = [c_{\text{A}}^{\text{s}}, c_{\text{B}}^{\text{s}}, T_{\text{r}}^{\text{s}}, T_{\text{c}}^{\text{s}}]^\intercal = [0.60, 3.62, 364.92, 349.13]^\intercal$. The dynamical behavior of CSTR is described by a non-linear model (2).

$$\frac{\text{d}c_{\text{A}}}{\text{d}t} = \frac{q_{\text{r}}}{V_{\text{r}}}(c_{\text{A0}} - c_{\text{A}}) - \nu_{\text{A}}\kappa c_{\text{A}}^{\nu_{\text{A}}} \tag{2a}$$

$$\frac{\text{d}c_{\text{B}}}{\text{d}t} = \frac{q_{\text{r}}}{V_{\text{r}}}(c_{\text{B0}} - c_{\text{B}}) + \nu_{\text{B}}\kappa c_{\text{A}}^{\nu_{\text{A}}} \tag{2b}$$

$$\frac{\text{d}T_{\text{r}}}{\text{d}t} = \frac{-H_{\text{r}}}{\rho_{\text{r}} C_{\text{r}}}\kappa c_{\text{A}} + \frac{q_{\text{r}}}{V_{\text{r}}}(T_{\text{r0}} - T_{\text{r}}) + \frac{hA}{V_{\text{r}}\rho_{\text{r}} C_{\text{r}}}(T_{\text{c}} - T_{\text{r}}) \tag{2c}$$

$$\frac{\text{d}T_{\text{c}}}{\text{d}t} = \frac{q_{\text{c}}}{V_{\text{c}}}(T_{\text{c0}} - T_{\text{c}}) + \frac{hA}{V_{\text{c}}\rho_{\text{c}} C_{\text{c}}}(T_{\text{r}} - T_{\text{c}}) \tag{2d}$$

$$\kappa = K_{\text{r}} e^{-\frac{E_{\text{r}}}{RT_{\text{r}}}} \tag{2e}$$

Table I provides the parameters of the model, along with their values and units. The model and parametrization were adopted from [16].

The sampling period of the system was chosen to be $T_{\text{s}} = 10\,\text{s}$.

B. Baseline Non-linear Model Predictive Control

The non-linear model predictive control (NMPC) serves as the baseline comparison with the quality of the control policy obtained by the DDPG algorithm. To make the formulation

of the NMPC relevant to the DDPG approach, we choose a similar objective function as in (1). Let,

$$\min_{U} \quad \sum_{k=0}^{N-1} (x_k^\intercal Q_\mathrm{x} x_k + u_k^\intercal Q_\mathrm{u} u_k) \tag{3a}$$

$$\text{s. t.} \quad x_{k+1} = f_\mathrm{m}(x_k, u_k), \quad k = 0, 1, \ldots, N-1, \tag{3b}$$

$$u_\mathrm{min} \leq u_k \leq u_\mathrm{max}, \quad k = 0, 1, \ldots, N-1, \tag{3c}$$

$$x_0 = x_\mathrm{init}, \tag{3d}$$

where the variable N is the prediction horizon, Q_x and Q_u are positive definite weighting matrices of appropriate size, defined with the same value as for the DDPG algorithm in (4). The model function $f_\mathrm{m}(\cdot)$ are discretized differential equations from (2). Here, the output of the optimization is the sequence of optimal control inputs, i.e., $U = [u_0^\star, \ldots, u_{N-1}^\star]^\intercal$, where only the first control input $u_0^\star$ is applied to the system.

The weighting matrices Q_x and Q_u were chosen such that the objective function and reward function gains similar contributions from individual state and input values. The input constraints u_min and u_max were set up with regards to the physical constraints of reactor. The NMPC problem was formulated with

$$Q_\mathrm{x} = \mathrm{diag}[1.669, 13.809, 0.274, 0.003], \tag{4a}$$

$$Q_\mathrm{u} = \mathrm{diag}[1.0, 0.1], \tag{4b}$$

$$N = 20, \tag{4c}$$

$$u_\mathrm{min} = [0.0, 0.0]^\intercal \mathrm{m}^3\,\mathrm{s}^{-1}, \tag{4d}$$

$$u_\mathrm{max} = [0.03, 0.008]^\intercal \mathrm{m}^3\,\mathrm{s}^{-1}, \tag{4e}$$

and was solved with the MATLAB 2023b Model Predictive Control Toolbox.

C. Training of Nearly Optimal Control Policy

The DDPG algorithm was deployed in the Reinforcement Learning Toolbox in MATLAB R2023b.

The approximation of the MPC with a neural network is a valid solution to the difficult task of implementing higher types of encrypted control. The neural network provides a suitable environment for encrypted calculations if executed correctly. One of the conditions that need to be met is that we need to use HE-friendly activation functions. Standard activation functions, such as the classical tangent, sigmoid, or ReLU, are not compatible with very limited set of arithmetic operations provided by HE schemes. This can be solved by using their polynomial approximations, rendering them HE-compliant. In this paper, we have opted for a polynomial approximation [17] of the hyperbolic tangent in the form $\tanh(x) \approx 0.0009x^5 - 0.0392x^3 + 0.6414x$.

Another limiting factor in the presented approach is the direct correlation between NN's structure and its multiplicative depth. The maximum number of consecutive multiplicative operations that ciphertexts are subjected to within the neural network dictates the number of levels of multiplicative depth. This is facilitated by choosing as many inner primes in an array of bit sizes for coefficient moduli $\mathcal{Q} = \mathcal{Q}_{S1} \times \mathcal{Q}_{\Delta 1} \times \cdots \times \mathcal{Q}_{\Delta M} \times \mathcal{Q}_{S2}$ as number of required consecutive multiplication. The designed NN (representing the actor network f_ACT) requires in total 9 multiplications in the chain (9 levels of multiplicative depth). More information about CKKS setup can be found in [5].

The f_ACT network consists of the input layer, two hidden layers with the HE-friendly activation function described above, and the output layer. The input layer has 4 neurons, corresponding to the state variables of the CSTR. Both hidden layers have 16 neurons, and the output layer has 2 neurons, corresponding to the control inputs. The f_CRIT network has a more complicated structure that is not the objective of this study. The f_CRIT network is present only in the training of the f_ACT, so the limitations concerning f_ACT are not imposed in the architecture of f_CRIT. Total, the f_ACT network has 38 neurons and the f_CRIT network has 79 neurons. Learning rate of f_ACT was set to $5 \cdot 10^{-5}$ and the learning rate of f_CRIT was set to $5 \cdot 10^{-4}$.

After creating the f_ACT and f_CRIT networks, we can proceed to training. During the training, the algorithm does not require any information about the model of the process. The only necessary information are obtained as the measured outputs of the system, the result of applied control actions by f_ACT. The training was performed in the close loop with the system as visualized in Figure 1. The training was carried out for 1000 simulations. In every simulation, the system started at the specific initial state. This initial state of the system in each simulation closely resembled the excited state of the system when subjected to a disturbance in the form of a changed inlet temperature of the reactant $T_{\mathrm{r}0}$.

The cumulative value of the cost function of this simulation was aimed to be minimized during training. This cumulative value was calculated as the sum of the cost function (1) over the whole simulation. The algorithm is design to maximize the reward function so negative value of (1) was calculated for the purpose of compatibility with the algorithm. The length of this simulation was specifically chosen to be the same as the prediction horizon of the NMPC controller to achieve similar results. Therefore, the length of each simulation was 200 s.

The training process took approximately 20 min and was carried out on a 4 cores of Apple M1 Pro processor. All parameters, hyperparameters, and the architectures of neural networks were chosen empirically and were tuned and adjusted based on the results of the training process.

D. Homomorphic Encryption in Close Loop

After obtaining trained control policy f_ACT as mentioned in V-C, we deployed the control law in the form of a Multi-Layer Perceptron (MLP) NN to control the CSTR. As shown in Fig. 3, the setup consists of process environment and cloud environment. The MLP NN control law is deployed in the plaintext form and process data are ciphertexts i.e. encrypted throughout the closed-loop roundtrip (network and NN evaluation). Keys are generated on the process side, and the public, relinearization, and Galois keys are exchanged between the process and the cloud at the beginning. States

are continuously measured, encrypted on the process side, and transmitted as ciphertexts over the network to the cloud. In the cloud, these ciphertexts are evaluated using the control law, which generates encrypted control inputs. These encrypted control inputs are sent back to the process side, where they are decrypted and applied to the system. This cycle repeats within the control loop. The specifics of the closed-loop implementation, deployment, and cryptographic setup are provided in Sec. VI.

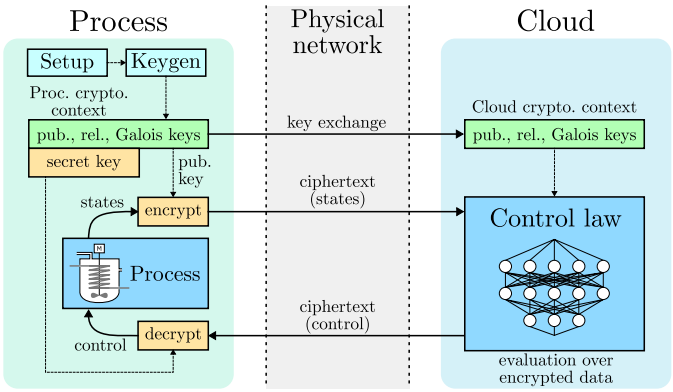

Fig. 3: The scheme of the encrypted control loop.

The objective was to control the CSTR to the origin represented by a steady state x^{s}. The system was subjected to the same type of disturbance as during the training phase, that is, to the change in the temperature of inlet flow T_{r0}. At time $T = 20\,\mathrm{s}$ the inlet flow decreases to $T_{\mathrm{r0}} = 310\,\mathrm{K}$ and at time $T = 40\,\mathrm{s}$ it increases back to $T_{\mathrm{r0}} = 325\,\mathrm{K}$. To test the robustness of the f_{ACT} we also introduced a second disturbance. This type of disturbance was not present during the training phase. The second disturbance was the failure of the cooling valve controlling q_{c}. This valve fails between times $150\,\mathrm{s}$ and $170\,\mathrm{s}$, is fully open and does not respond to the control signal.

The results of this simulation are visualized in Figure 4 and 5. The CSTR was simulated with the sampling time $T_{\mathrm{s}} = 0.25\,\mathrm{s}$ using an *ode45* solver in MATLAB R2023b.

E. Comparison of Encrypted DDPG with NMPC in Performance

The comparison of the algorithms was done in simulation, where 20 different random disturbances occurred, 10 on the inlet temperature of the reactant T_{r0} and 10 on the inlet temperature of the coolant T_{c0}. Disturbances lasted for $20\,\mathrm{s}$. The results of the comparison are presented in the following section (Section VI). The comparison was done by evaluating integral square error (ISE) criteria for both states and control inputs. The ISE criteria were calculated as the sum of the squared values of states and control input:

$$\mathrm{ISE_x} = \sum_{i=1}^{N_{\mathrm{test}}} (x_i)^2\,, \qquad \mathrm{ISE_u} = \sum_{i=1}^{N_{\mathrm{test}}} (u_i)^2\,, \tag{5}$$

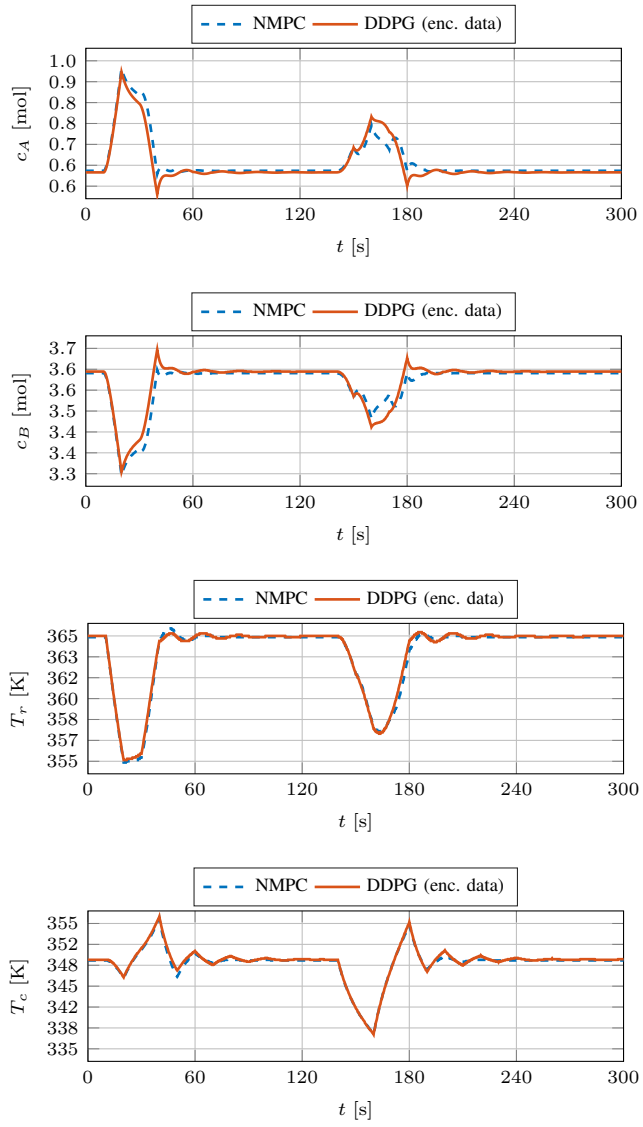

Fig. 4: Control trajectories of CSTR states with NMPC (blue dashed) and encrypted DDPG controller (red solid).

where x_i and u_i are the state and control input values at time step i, respectively. The N_{test} is the length of the simulation during which the random disturbances occurred.

TABLE II: Comparison of NMPC and DDPG with encrypted data controllers

Algorithm	NMPC	DDPG (enc. data)
$\mathrm{ISE}_{c_{\mathrm{A}}}$	1.466	1.338
$\mathrm{ISE}_{c_{\mathrm{B}}}$	1.466	1.338
$\mathrm{ISE}_{T_{\mathrm{r}}}$	$1.937 \cdot 10^{3}$	$1.890 \cdot 10^{3}$
$\mathrm{ISE}_{T_{\mathrm{c}}}$	$4.420 \cdot 10^{2}$	$5.423 \cdot 10^{2}$
$\mathrm{ISE}_{q_{\mathrm{r}}}$	$3.272 \cdot 10^{-4}$	$4.209 \cdot 10^{-4}$
$\mathrm{ISE}_{q_{\mathrm{c}}}$	$3.447 \cdot 10^{-4}$	$3.693 \cdot 10^{-4}$

The results of the comparison are presented in Table II. The average performance degradation of ISE criteria of DDPG

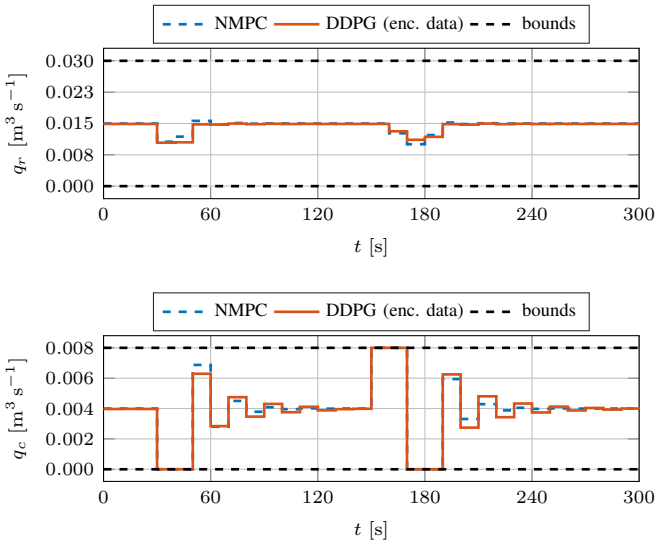

Fig. 5: Trajectories of CSTR control actions with NMPC (blue dashed) and encrypted DDPG controller (red solid).

over NMPC is 6.46%. This indicates that the NMPC controller provides better control performance compared to the DDPG controller in the presence of random disturbances. However, the difference in performance is small. Regarding many benefits of DDPG control over NMPC, the results indicate that the DDPG controller with encrypted data is a promising safe alternative to NMPC.

VI. Deployment Benchmarks

In this section, we discuss a deployment setup of an encrypted control. In the majority of pertinent literature, evaluations of homomorphically encrypted controls are typically conducted through simulation within a local environment, often executed as a standalone computer program. Consequently, certain challenges inherent in practical implementations may be overlooked. In our work, we still use a simulation model as a substitute for a controlled process, however we ensure that all other components of the control loop are implemented to accurately emulate real-world deployment. The setup consists of two virtual machines (VM) running on a host computer via a Hyper-V virtualization. The host computer uses an AMD Ryzen 9 5950X CPU with 16 cores and 32 threads, is equipped with 128 GB of DDR4 RAM clocked at 3200 MHz, and uses a 10 Gbit Ethernet interface, integrated into the local physical network, which adheres to the same specifications. For each VM, we allocated 4 physical CPU cores and 16 GB of RAM.

The first VM represents the process side of a control loop. It runs a simulation model of a controlled process implemented in MATLAB. The process exchanges the data with a program written in Python that provides cryptographic features and serves as a communication bridge to the cloud. The second VM represents the cloud side of the control loop. The cloud logic is implemented in Python and contains the outsourced control law (the neural network designed in Sec. V-C). The communication between the two is handled via HTTPS, and data is transferred using the JSON format with hexadecimal encoding. To utilize the full communication stack, we route the traffic through a local network, which is a more realistic scenario than just the use of a direct loopback adapter that does not utilize the physical layer.

In this benchmark, we perform an encrypted closed-loop control of the reactor model and simultaneously measure the time of all important steps as well as the memory footprint of data associated with them. The procedure consists of two phases, the first being the handshake, where the cryptographic keys are generated on the side of the process and sent to the cloud, and the second being the actual control loop runtime. We use the CKKS cryptosystem provided by the Python library TenSEAL [18] that is built on top of the Microsoft SEAL [19]. The cryptographic parameters used in this setup are: the polynomial modulus degree $N = 16384$; scaling factor $\Delta = 37\,\text{bit}$; and coefficient moduli chain $\mathcal{Q} = [45\,\text{bit}, 9 \times 37\,\text{bit}, 45\,\text{bit}]$.

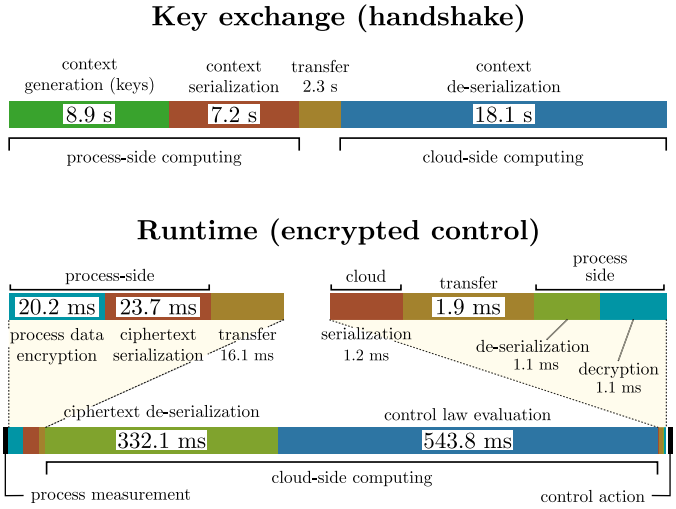

Fig. 6: Time-based visualization of cryptographic handshake and encrypted control runtime.

The average times of both the key exchange and runtime are visualized in Fig. 6. The generation of cryptographic context, the memory object (in binary format – BIN) that holds keys and parameters, takes on average 8.9 s. Then the context that contains all the keys, except the private key, is serialized into a hexadecimally encoded string – HEX (7.2 s) and sent to the cloud (2.3 s). The size of the context takes up 549.2 MB (BIN) in the memory of process-side computer, 548.2 MB (BIN) of a cloud computer, and 1.07 GB (HEX) for transfer. The cloud deserializes the context (18.1 s), and after the acknowledgment, the handshake is complete. The runtime consists of an iterative procedure where the process states are measured, encoded/encrypted (20.2 ms, 1.81 MB – BIN) and sent to the cloud (16.1 ms). The cloud computer deserializes the ciphertext (332.1 ms), and evaluates the control law (543.8 ms) to get an encrypted control action (215.16 kB – BIN). The resulting ciphertext is then serialized and sent back

to the process (1.9 ms). The process deserializes the ciphertext (1.13 ms), decrypts/decodes it (1.05 ms) and applies it to the process.

The sizes of all the keys and ciphertexts are presented in Table III. The times involved in the key exchange and runtime are presented in Tables IV and V, respectively. The times and sizes were acquired from 60 repetitions of the key exchange and 60 iterations of the control loop. The presented data show that the cryptographic handshake takes on average 36.5 s, and a single iteration of the control loop runtime takes on average 941 ms, which is well below the sampling time of the control loop. The major benefit of implemented controller comes from its fixed structure, thus very consistent computational times, unlike the baseline NMPC that on average took 3.09 s with minimum of 0.32 s and maximum of 14.7 s.

TABLE III: Sizes of cryptographic keys and ciphertexts, both in binary (memory utilization) and hexadecimal (transfer).

Size of (p–process, c–cloud)	BIN	HEX
Public key (p, c)	2.02 MB	4.04 MB
Secret key (p)	1.01 MB	2.02 MB
Relinearization key (p, c)	20.23 MB	40.46 MB
Galois key (p, c)	526 MB	1.03 GB
Ciphertext (states, level 9)	1.81 MB	3.62 MB
Ciphertext (control, level 0)	215.16 kB	430.32 kB

TABLE IV: Benchmark results for key exchange (handshake), including the times[1] for key generation, serialization, deserialization, and network transfer. Presented data were acquired from 60 repetitions of a key exchange.

Time of [seconds] (p–process, c–cloud)	Min.	Max.	Mean	Median
Key generation (p)	8.77	9.06	8.88	8.88
Serialization (p)	7.13	7.50	7.24	7.22
Deserialization (c)	16.7	18.9	18.1	18.1
Network transfer	2.31	2.33	2.34	2.34
Round-trip (tot.)	34.9	37.8	36.5	36.5

VII. Conclusion

In this study, we explored the integration of the Deep Deterministic Policy Gradient (DDPG) with homomorphic encryption control over the model of a continuous stirred tank reactor. Our approach combines the strength of DDPG for generating explicit control laws in the form of neural networks (NNs) and the privacy preservation capabilities of HE. The application of polynomial approximations for activation functions within the NNs aligns with the constraints of HE, enabling secure and private control. Through a detailed experimental setup, including deployment benchmarks in a simulated real-world environment, we demonstrate the practical viability of this approach. Our findings indicate that while the implementation of homomorphically encrypted control systems introduces computational overheads, it ultimately outweighs its shortcomings and offers a valid, worthwhile solution. This study paves the way for future research in encrypted control systems, highlighting the balance between control performance and data privacy in cloud-based control applications.

[1]Note that the utilized time measurement method has a limited accuracy of 318 µs, which mostly affects the measurements of short time periods.

TABLE V: Benchmark results for the runtime of encrypted control, including the times[1] for encryption, decryption, serialization, deserialization, total round-trip, network transfer, and control law evaluation. The data presented was acquired from 60 iterations of the control loop.

Time of [milliseconds] (p–process, c–cloud)	Min.	Max.	Mean	Median
Encode/Encrypt (p)	19.0	23.0	20.2	20.0
Decode/Decrypt (p)	1.00	2.00	1.05	1.00
Serialization (p)	23.0	30.0	23.7	24.0
Serialization (c)	1.00	2.00	1.15	1.00
Deserialization (p)	1.00	2.00	1.13	1.00
Deserialization (c)	62.0	381	332	363
Network transfer (tot.)	17.0	23.0	18.3	18.0
Evaluation (c)	531	567	544	544
Round-trip (tot.)	661	1009	941	969

References

[1] E. D. Knapp and J. T. Langill, "Chapter 3 - industrial cyber security history and trends," in *Industrial Network Security (Second Edition)*, second edition ed., E. D. Knapp and J. T. Langill, Eds. Boston: Syngress, 2015, pp. 41–57. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124201149000034

[2] M. Wolf, "Chapter 8 - cyber-physical systems," in *High-Performance Embedded Computing (Second Edition)*, second edition ed., M. Wolf, Ed. Boston: Morgan Kaufmann, 2014, pp. 391–413. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124105119000083

[3] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A survey on scada systems: Secure protocols, incidents, threats and tactics," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1942–1976, 2020.

[4] M. S. Darup, A. B. Alexandru, D. E. Quevedo, and G. J. Pappas, "Encrypted control for networked systems: An illustrative introduction and current challenges," *IEEE Control Systems Magazine*, vol. 41, no. 3, pp. 58–78, 2021.

[5] M. Furka, M. Kalúz, M. Fikar, and M. Klaučo, "Guidelines for secure process control: Harnessing the power of homomorphic encryption and state feedback control," *IEEE Access*, vol. 11, pp. 110 328–110 341, 2023.

[6] M. S. Darup, "Encrypted polynomial control based on tailored two-party computation," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 11, pp. 4168–4187, 2020.

[7] M. Furka, K. Kiš, M. Klaučo, and M. Kvasnica, "Usage of homomorphic encryption algorithms in process control," in *2021 23rd International Conference on Process Control (PC)*, 2021, pp. 43–48.

[8] N. Schlüter and M. S. Darup, "Encrypted explicit mpc based on two-party computation and convex controller decomposition," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 5469–5476.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[10] S. Kosieradzki, X. Zhao, H. Kawase, Y. Qiu, K. Kogiso, and J. Ueda, "Secure teleoperation control using somewhat homomorphic encryption," *IFAC-PapersOnLine*, vol. 55, no. 37, pp. 593–600, 2022, 2nd Modeling, Estimation and Control Conference MECC 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322028907

[11] K. Tjell, N. Schlüter, P. Binfet, and M. S. Darup, "Secure learning-based mpc via garbled circuit," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 4907–4914.

[12] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds. Cham: Springer International Publishing, 2017, pp. 409–437.
[13] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.
[14] Y. Lin, J. McPhee, and N. L. Azad, "Comparison of deep reinforcement learning and model predictive control for adaptive cruise control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 221–231, 2020.
[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[16] M. Bakošová, A. Mészáros, J. Klemeš, and J. Oravec, "Robust and optimal control approach for exothermic reactor stabilization," *Theoretical Foundations of Chemical Engineering*, no. 46, pp. 740–746, 2012.
[17] M. Kalúz, R. Kohút, and D. Dzurková, "Mpc-mimicking neural network based on homomorphic encryption," in *Proceedings of the 2023 24th International Conference on Process Control*, Slovak University of Technology in Bratislava. IEEE, June 6 - 9, 2023 2023, pp. 126–131.
[18] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "Tenseal: A library for encrypted tensor operations using homomorphic encryption," https://arxiv.org/abs/2104.03152, 2021.
[19] "Microsoft SEAL (release 4.1)," https://github.com/Microsoft/SEAL, Jan. 2023, Microsoft Research, Redmond, WA.