MPC-Mimicking Neural Network Based on Homomorphic Encryption

Martin Kalúz, Roman Kohút, and Diana Dzurková
Faculty of Chemical and Food Technology
Slovak University of Technology in Bratislava
Bratislava, Slovak Republic
{martin.kaluz, roman.kohut, diana.dzurkova}@stuba.sk

***Abstract*—This paper showcases the use of homomorphic encryption (HE) scheme for securing process data during the controller evaluation in a simulated untrusted cloud environment. The controller implemented in this work is a neural network (NN) that mimics a model predictive controller (MPC) designed for disturbance rejection. Firstly, an MPC was designed for a process of biochemical reactor. From obtained MPC control data, a neural network (NN-MPC) with fully connected layers was trained. Multiple HE-friendly activation functions were tested during the NN training and testing, and based on the results, a polynomial approximation of hyperbolic tangent was used. Subsequently, the NN-MPC controller was implemented in encrypted control scenario. The measured states of the biochemical reactor were encrypted on the side of the process and sent for the homomorphic evaluation to the simulated cloud (NN-MPC).**

***Index Terms*—Model Predictive Control, Neural Network, Approximation, Biochemical Reactor, Process Control, Homomorphic Encryption, Data Privacy, CKKS**

I. Introduction

With control algorithms' growing computational complexity and processing data size, the demand for cloud-based services has become more relevant in recent years. Computing on the cloud can be very beneficial for the data owner since it takes away the computational and data storage demands and outsources them to other parties. However, these benefits come with risks regarding the security and privacy of the data. Most encryption standards, such as the AES and RSA, focus on data security during the transfer. These data are decrypted on the cloud computer before they are processed, raising the concern of data privacy violations. One plausible solution is to use homomorphic encryption (HE), allowing the outsourced algorithms to process the data in encrypted form.

The literature provides numerous examples of HE-enabled control setups [1] with many open challenges [2]. HE is being implemented in various control applications, using polynomial controllers [3], linear feedback [4], and non-linear controllers [5]. Some more advanced control scenarios, like implicit and explicit MPC, were showcased in [6], [7].

In the past decade, several new cryptosystems emerged, primarily based on the ring version of the Learning with Errors (LWE/RLWE) problem [8]. In this paper, we use one of them, the CKKS cryptosystem.

With the security and data-privacy benefits of HE schemes also come some drawbacks. In HE, only a limited set of operations is usually available (mainly addition and multiplication). Also, the computational complexity and memory demands are much higher than for unencrypted applications. These facts pose a significant challenge in implementing complex controllers like MPC, requiring optimization over encrypted data. These problems can be tackled by approximating control law with more HE-friendly constructs such as neural networks (NN). In [9], a linear MPC was implemented as a non-polynomial max-out neural network with a single hidden layer.

This paper presents a NN approximation of linear MPC (NN-MPC) designed for state disturbance rejection to control the biochemical reactor model with input constraints. Used NN-MPC consists of 3 fully connected hidden layers (50 neurons each) and an output layer. The inference of NN-MPC is performed over encrypted state measurements of the process, yielding an encrypted control action.

II. CKKS Cryptosystem

CKKS (Cheon-Kim-Kim-Song) cryptosystem is an RLWE-based scheme that operates with approximated fixed-point arithmetic, originally published under the name "*Homomorphic Encryption for Arithmetic of Approximate Numbers*" (HEAAN) [10]. CKKS is one of the most implemented schemes in modern HE, included in libraries and frameworks such as OpenFHE[1], Palisade[2], HElib[3], Lattigo[4], and Microsoft SEAL[5]. It classifies as leveled HE scheme, meaning that additions and multiplication between two encrypted messages are possible, but the number of consecutive multiplications is limited. This limitation comes from a noise (additive error) incorporated into encrypted messages to make them cryptographically secure, which is the main idea behind the hardness assumption of RLWE [8].

In the following sections II-A–II-D, we describe the basic principles of the CKKS cryptosystem. Detailed inner workings of the mathematics behind the scheme can be found in [10], [11].

A. Notation for Message, Plaintext and Ciphertext Spaces

In this paper, we use the following notation and terms. The CKKS operates over polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N+1)$,

[1]https://github.com/openfheorg/openfhe-development
[2]https://palisade-crypto.org
[3]https://github.com/homenc/HElib
[4]https://github.com/tuneinsight/lattigo
[5]https://www.microsoft.com/en-us/research/project/microsoft-seal/

where q is a coefficient modulus, and (X^N+1), for N being a power of two, represents irreducible cyclotomic polynomial (basically the equivalent of modulus in polynomial space). N being a polynomial modulus degree defines a maximum size of polynomials in the ring $\mathcal{R}_q$ to be $(N-1)$.

The raw input message $m \in \mathbb{C}^{\frac{N}{2}}$ is a vector of $\frac{N}{2}$ complex numbers, of which usually only real parts are used in practical applications. Plaintext $m' \in \mathcal{R}_q$ is a polynomial with N coefficients, obtained by encoding procedure $\mathcal{E} : m \in \mathbb{C}^{\frac{N}{2}} \rightarrow m' \in \mathcal{R}_q$ (Sec. II-B). The ciphertext $c = (c_1, c_2) \in \mathcal{R}_q^2$ is a tuple of two polynomials obtained by encryption procedure $E : m' \in \mathcal{R}_q \rightarrow c \in \mathcal{R}_q^2$, described in section II-C.

B. Encoding and Decoding

In the CKKS, the encoding function $\mathcal{E}(\cdot)$ is defined as

$$m' = \mathcal{E}(m, \Delta) = \left[\Delta \cdot \pi^{-1}(m)\right], \tag{1}$$

where a complex canonical embedding function $\pi(\cdot)$ maps the coefficients of polynomial m' into the elements of m. Symbol Δ represents a scaling factor that multiplies polynomial coefficients to move the bits of encoded message to the left, creating a space for addition of cryptographic noise during the encryption without the significant loss of numerical precision. The mathematical details of embedding and encoding are provided in [10, section 2.2 and 3.2]. The decoding procedure $\mathcal{D} : m' \in \mathcal{R}_q \rightarrow m \in \mathbb{C}^{\frac{N}{2}}$ is just an inverse of (1), defined as

$$m = \mathcal{D}(m', \Delta) = \pi\left(\frac{1}{\Delta} \cdot m'\right). \tag{2}$$

It is clear from (1) that rounding function $[\cdot]$ will result in only approximated decoding of original message $\mathcal{D}\left(\mathcal{E}(m, \Delta), \Delta\right) \approx m$, hence the name of the scheme "*Homomorphic Encryption for Arithmetic of Approximate Numbers*".

C. Key Generation, Encryption and Decryption

CKKS is an asymmetric scheme. One party generates a secret key and a set of public keys for distribution. A secret polynomial $s \in \mathcal{R}$ with signed binary coefficients $\{-1, 0, 1\}$ is sampled from a distribution $\mathcal{HWT}(h)$ described in [10, section 3.4]. In practical implementation, the secret key is a tuple $sk = (1, s)$. The public key $pk = (pk_1, pk_2)$ is a tuple of two polynomials

$$pk_1 = [-a \cdot sk + e]_q, \tag{3}$$
$$pk_2 = a \tag{4}$$

where polynomial a is uniformly sampled from $\mathcal{R}_q$ and $e \hookleftarrow \mathcal{X}$ is an error polynomial with coefficients sampled from discrete Gaussian distribution $\mathcal{X}$ (see [10, section 2.3]).

The encryption of plaintext m' is done by using a public key pk and forming a ciphertext $c = (c_1, c_2) \in R_q^2$ such that

$$c_1 = [pk_1 \cdot u + e_1 + m']_q, \tag{5}$$
$$c_2 = [pk_2 \cdot u + e_2]_q, \tag{6}$$

where $u \in \mathcal{R}_q$ is a random polynomial and e_1, e_2 are error polynomials from $\mathcal{X}$. Decryption of c is performed by evaluating a ciphertext over secret polynomial s

$$\tilde{m}' = [c_1 + c_2 \cdot s]_q \tag{7}$$

to obtain a plaintext with approximation $\tilde{m}'$ of originally encrypted m'.

D. Homomorphic Operations

CKKS allows operations to be carried out between two ciphertext or between plaintext and ciphertext. In fact, the operations are just a standard polynomial algebra over the ring $\mathcal{R}_q$. Considering two ciphertexts $c_\mathrm{a} = (c_{\mathrm{a},1}, c_{\mathrm{a},2})$ and $c_\mathrm{b} = (c_{\mathrm{b},1}, c_{\mathrm{b},2})$ with the same value of the modulus q, we can calculate the product of their homomorphic addition c_add as

$$c_{\mathrm{add},1} = [c_{\mathrm{a},1} + c_{\mathrm{a},2}]_q, \tag{8}$$
$$c_{\mathrm{add},2} = [c_{\mathrm{b},1} + c_{\mathrm{b},2}]_q. \tag{9}$$

The homomorphic multiplication between ciphertext c and plaintext k' gives a new ciphertext

$$c_\mathrm{pmul} = k' \cdot c = ([k' \cdot c_1]_q, [k' \cdot c_2]_q), \tag{10}$$

and for two ciphertexts, multiplication between c_a and c_b is performed as

$$c_\mathrm{cmul} = c_\mathrm{a} \cdot c_\mathrm{b} = (c_\mathrm{cmul1}, c_\mathrm{cmul2}, c_\mathrm{cmul3}), \tag{11}$$
$$c_\mathrm{cmul1} = \left[c_\mathrm{a1} \cdot c_\mathrm{b1}\right]_q, \tag{12}$$
$$c_\mathrm{cmul2} = \left[c_\mathrm{a1} \cdot c_\mathrm{b2} + c_\mathrm{b1} \cdot c_\mathrm{a2}\right]_q, \tag{13}$$
$$c_\mathrm{cmul3} = \left[c_\mathrm{a2} \cdot c_\mathrm{b2}\right]_q. \tag{14}$$

It is clear that by multiplying two ciphertext, the result c_cmul will grow in size. Therefore, it is necessary to perform a re-linearization

$$\begin{aligned} c'_\mathrm{cmul} &= (c'_\mathrm{cmul1}, c'_\mathrm{cmul2}) \\ &= (c_\mathrm{cmul1}, c_\mathrm{cmul2}) + [p^{-1} \cdot c_\mathrm{cmul3} \cdot rk], \end{aligned} \tag{15}$$

where rk is a re-linearization key and p is a big integer.

One of the main features of CKKS is message batching, which allows inclusion of multiple message elements into the slots of single plaintext or ciphertext. For polynomial modulus degree N (usually in thousands), $N/2$ individual numbers can be included in one plaintext/ciphertext. All the slots are evaluated simultaneously during one homomorphic operation, bringing the considerable potential for applications where parallelism is desired. Additionally, CKKS implementations in HE frameworks also provide a vector rotation technique. During this procedure, plaintext/ciphertext slots are shifted and wrapped around (either to the left or right). This technique is used to implement homomorphic multiplication between plaintext matrices and ciphertext vectors, which is especially useful in evaluating layers in NNs. The main drawback of HE vector rotation is that it requires a set of cryptographic keys (Gallois keys) that are large.

III. NN-MPC Disturbance Rejection Control of Biochemical Reactor over Encrypted Data

A. Process Description

We consider a biochemical reactor with an alcoholic fermentation process. During the reactor operation, microorganisms convert saccharides such as fructose or glucose into ethanol and carbon dioxide. This complex biochemical transformation creates many by-products. However, the overall reaction can be summarized as follows:

$$C_6H_{12}O_{6(s)} \rightarrow 2C_2H_5OH_{(l)} + 2CO_{2(g)}, \quad \Delta H_r < 0. \tag{16}$$

Besides saccharides concentration, the temperature's influence must be considered to maintain ideal microorganisms' growth. From [12] optimal temperature for the presented yeast growth rate was determined to be between $28\ ^{\circ}C - 32\ ^{\circ}C$. As the aeration inhibits saccharide consumption, the considered model includes oxygen concentration as one of the state variables.

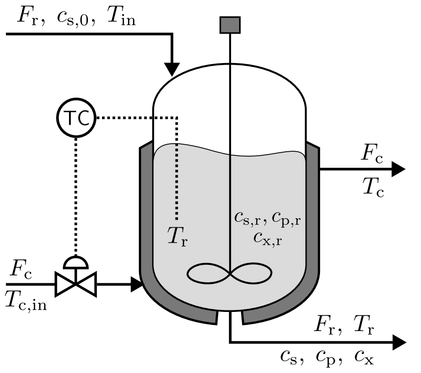
$F_\mathrm{r},\ c_\mathrm{s,0},\ T_\mathrm{in}$
TC
F_c
T_c
T_r
$c_\mathrm{s,r}, c_\mathrm{p,r}$
$c_\mathrm{x,r}$
F_c
$T_\mathrm{c,in}$
$F_\mathrm{r},\ T_\mathrm{r}$
$c_\mathrm{s},\ c_\mathrm{p},\ c_\mathrm{x}$

Fig. 1. Schematic diagram of biochemical reactor

The biochemical reactor is modeled as a continuous stirred tank reactor Fig. 1. The tank is fed with a constant glucose solution $F_r = 25\ l/h$, feeding the biomass (suspension of yeasts) and producing the continuous outlet flow of ethanol. The processes in the reactor are described with six differential equations obtained along with their parameters from [13].

B. Model Predictive Control for Disturbance Rejection

The presented model predictive control (MPC) takes the form of output steady-state control with discrete time state space model and Δu penalization:

$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} \left(y_k^\top Q y_k + \Delta u_k^\top R \Delta u_k \right) \tag{17}$$

$$\text{s.t.} \quad x_{k+1} = A_\mathrm{d} x_k + B_\mathrm{d} u_k, \tag{18}$$

$$y_k = C_\mathrm{d} x_k, \tag{19}$$

$$\Delta u_k = u_k - u_{k-1}, \tag{20}$$

$$x_{k+1} \in \mathcal{X}, u_k \in \mathcal{U}, \Delta u_k \in \mathcal{U}_\mathrm{du} \tag{21}$$

$$x_0 = x(t), u_{-1} = u(t - Ts), \tag{22}$$

$$k = 0, 1, \dots, N-1, \tag{23}$$

where N is finite prediction horizon, $Q \succeq 0, R \succeq 0$ are penalty matrices. Vectors $x_k \in \mathbb{R}^{n_\mathrm{x}}$, $u_k \in \mathbb{R}^{n_\mathrm{u}}$, $y_k \in \mathbb{R}^{n_\mathrm{y}}$ represents system state, input, and controlled output predictions for define step k, respectively. Model presented includes state matrix $A_\mathrm{d} \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$, input matrix $B_\mathrm{d} \in \mathbb{R}^{n_x \times n_u}$, and output matrix $C_\mathrm{d} \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{y}}$. $\mathcal{X} \in \mathbb{R}^{n_\mathrm{x}}$, $\mathcal{U} \in \mathbb{R}^{n_\mathrm{u}}$, $\mathcal{U}_\mathrm{du} \in \mathbb{R}^{n_\mathrm{u}}$ are polyhedral sets of constraints for states, control input and change of the control input respectively.

The linear discrete time model Eq. (18) was derived from the non-linear model of the biochemical reactor presented in [13]. System states include $[c_\mathrm{x}, c_\mathrm{p}, c_\mathrm{s}, c_{\mathrm{O}_2}, T_\mathrm{r}, T_\mathrm{c}]^\top$, biomass, ethanol, glucose, dissolved oxygen concentrations, reactor temperature, and the coolant's temperature subsequently. The state-space matrices are formulated as follows:

$$A_\mathrm{d} = \begin{bmatrix} 0.99 & -0.004 & 0 & 0 & 0.002 & 0 \\ 0.31 & 0.95 & 0.001 & 0 & 0 & 0 \\ -0.75 & 0.063 & 0.97 & 0 & 0 & 0 \\ -0.02 & 0 & 0 & 0 & -0.07 & -0.002 \\ 0.23 & 0 & 0 & 0.001 & 0.95 & 0.025 \\ 0.07 & 0 & 0 & 0 & 0.53 & 0.39 \end{bmatrix} \tag{24}$$

$$B_\mathrm{d} = [0\ \ 0\ \ 0\ \ 0.2\ \ -2.2\ \ -97.9]^\top \cdot 10^{-3} \tag{25}$$

$$C_\mathrm{d} = [0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0]^\top \tag{26}$$

where the sampling period is $T_\mathrm{s} = 0.5\ h$, and steady-state values for the ideal yeast growth environment $[c_\mathrm{x}^s, c_\mathrm{p}^s, c_\mathrm{s}^s, c_{\mathrm{O}_2}^s, T_\mathrm{r}^s, T_\mathrm{c}^s]^\top = [0.92\ g/l, 12.6\ g/l, 29.5\ g/l, 6.02\ mg/l, 29.6\ ^{\circ}C, 27.1\ ^{\circ}C]^\top$ and $F_c^s = 18\ l/h$ are selected as the linearization point.

The MPC's goal is to stabilize the biochemical reactor at its steady state and eliminate the disturbance's effect to provide ideal conditions for stable ethanol production. The controlled variable is the temperature in the reactor T_r, as the microorganism growth is linked to it. The manipulated variable is the coolant flow F_c. The penalty matrices are selected as $Q = 1000$ and $R = 1$. The prediction horizon is set to $N = 10$. The manipulated variable's constraints are $-18 \leq u \leq 162$, and the change of the manipulated variable is constrained to $-18 \leq \Delta u \leq 9$. The state constraints $[-1.11, -13.89, -26.23, -5.83, -12.16, -20.22]^\top \leq x$ are presented only for safety reasons. They can not be reached during normal reactor operations. Note that the variables and constraint values are stated in the deviation form.

C. Neural Network Approximation

The main drawback of RLWE-based HE schemes is that they are not usable for evaluating complex algorithms such as MPC. Therefore, we decided to approximate the MPC controller by a neural network (NN-MPC) with an appropriate structure that would allow us to perform a homomorphic inference over encrypted states to compute an encrypted control action. To obtain a good approximation of the MPC controller (17–23), we set up a series of experiments (66 in total) with different initial state conditions, generated by quantizing the state space, and performed MPC control to steady state. Overall 9823 data samples were obtained, containing six state variables and corresponding MPC control action for each sample. Before the NN training, the state variables were normalized. The data samples were then randomly shuffled, and 70% was used as a training set for NN-MPC. The remaining samples were split into two sets of the same size and used as validation and testing data. The NN was trained in Python deep learning API Keras, using an Adam Optimizer. The learning rate was set to 2.5×10^{-4}, and we used a mean square error as a loss function. The number of training epochs was set to 2500.

With respect to the maximum multiplicative depth of the CKKS setup (Fig. 2), we have chosen a neural network with three fully connected layers, 50 neurons each, and a linear output layer. The main limitation of HE schemes is that

TABLE I
Comparison of activation functions used in MCP-mimicking NN.

Activation name	Activation function	Prediction error (RMSE)	No. of strict constr. violations (upper/lower)	No. of constr. violations $> 0.1\%$ (upper/lower)	Severity of constr. violations (RMSE) (upper/lower)	Implementable in HE	HE mult. depth (levels)
tanh	$\frac{e^{2x}-1}{e^{2x}+1}$	0.148	114 / 80	0 / 0	0.028 / 0.047	no	–
sigmoid	$\frac{1}{1+e^{-x}}$	0.398	119 / 4	32 / 3	0.380 / 0.235	no	–
ReLU	$\max(0,x)$	0.568	75 / 72	22 / 41	0.648 / 0.372	no	–
p-approx. tanh	$-0.891x^3+1.336x^2$ $+0.594x-0.0196$	0.769	19 / 78	14 / 65	1.352 / 1.042	yes	2
p-approx. sigmoid	$-0.00219x^3+0.164x^2$ $+0.5$	1.599	28 / 66	25 / 53	0.844 / 1.444	yes	2
p-approx. ReLU	$-0.002x^4+0.147x^2$ $+0.5x+0.12$	1.500	64 / 76	58 / 59	1.273 / 0.620	yes	3
square	x^2	2.186	31 / 54	27 / 51	4.223 / 3.916	yes	1

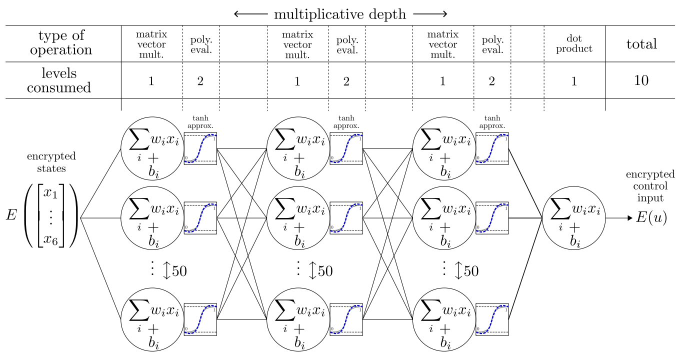

Fig. 2. A structure of implemented MPC-mimicking neural network with multiplicative depths of layers and activation functions

they are unsuitable for evaluating arbitrary functions such as those used in activations in NNs. To overcome this issue, we tested several most commonly used homomorphic approximations of activation functions [14]. These are polynomial approximations of hyperbolic tangent (tanh), sigmoid, ReLU, and a commonly used HE-friendly square function. Table I compares four approximated activations along with classical tanh, sigmoid, and ReLU. In this comparison, we focused on the root mean square error (RMSE) value between NN testing data and NN prediction, the number of strict and non-strict constraint violations, the RMSE of those violations, and the HE multiplicative depth of a function. The results show that all NNs with approximated activation functions provide worse prediction performance than NNs with standard activations. From all the activations implementable in HE, the smallest prediction error was achieved by approximated tanh function (p-approx. tanh). Figure 3 shows the prediction performance of trained NN-MPC using a p-approx. tanh as an activation function. It is clear that control action occasionally violates the constraints, however, these violations are just minor with RMSE value of 1.352 for upper and 1.042 for lower constraints, and overall RMSE prediction error only 0.769 (Tab. I).

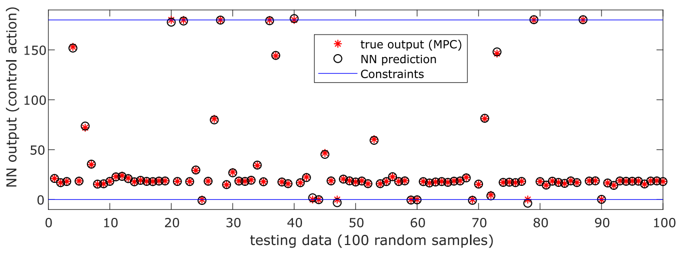

Fig. 3. Trained neural network performance illustrated on 100 randomly chosen data samples from testing set.

D. Homomorphic Multiplicative Depth of NN

In RLWE-based HE schemes, one of the most critical aspects of computation is the depth of the arithmetic circuit. Every ciphertext contains cryptographic noise (in the case of CKKS, also an approximation error) that grows in size with the number of operations being carried out over it. This will eventually lead to the corruption of encrypted data if noise reaches a threshold known as noise budget. To avoid the exponential growth of error during the ciphertext multiplication, CKKS employs a modulus-switching technique called rescaling. However, this procedure can be performed a limited number of times and is dictated by the number of inner primes Γ_i in coefficient moduli array (Sec. III-E).

The structure of NN used in this work (Fig. 2) contains a chain of consecutive ciphertext multiplications called multiplicative depth. For each hidden NN layer, the plaintext matrix of weights is homomorphically multiplied by an encrypted vector of states. The output layer also requires one ciphertext multiplication. In CKKS, the plaintext polynomial evaluation over ciphertext is of depth $n-1$, for n being an order of the polynomial. In the case of p-approx. tanh activation function, the multiplicative depth is two. Overall the depth of NN is 10.

E. Setup of Cryptographic Parameters

Several essential parameters (Table II) are considered when setting up the cryptosystem. First is polynomial modulus degree N. Bigger N increases the security of the scheme but, also increases the computational complexity. Simultaneously, larger N allows for a higher multiplicative depth of the arithmetic circuit. This is done by selecting an array of bit sizes for coefficient moduli $[\mathit{\Gamma}_\text{o}, \mathit{\Gamma}_\text{i}, \cdots \mathit{\Gamma}_\text{i}, \mathit{\Gamma}_\text{o}]$. The difference between the outer primes $\mathit{\Gamma}_\text{o}$ and inner primes $\mathit{\Gamma}_\text{i}$ (in bits) defines the bit precision of integer parts of encoded/encrypted numbers (in our case 17 bits). The number of inner primes $\mathit{\Gamma}_\text{i}$ (10) defines the maximum multiplicative depth. The scale Δ, usually chosen to be the same value as $\mathit{\Gamma}_\text{i}$, controls numerical precision after the decimal point of encoded/encrypted numbers, such that the precision is roughly the difference between the inner prime bit size and precision before the decimal point (in our case $33 - 17 = 16$ bits).

TABLE II
CKKS PARAMETERS USED IN THE CASE STUDY.

Parameter	Symbol	Value	Impacts
Polynomial modulus degree	N	16384	security level, mult. depth, performance
Coefficient mod. bit sizes	$\mathit{\Gamma}$	$[50, 33, 33, 33, 33, 33$ $33, 33, 33, 33, 33, 50]$	mult. depth, precision
Scale	Δ	33	precision

F. Control over Encrypted Data

The control setup consists of three environments. First is MATLAB, where numerical simulation of the biochemical reactor is performed using `ode23s` solver. MATLAB communicates with an encryption/decryption layer written in Python, where measured states of the process are sent to be encrypted, and encrypted control actions from NN-MPC are decrypted before they are sent back to the process. This layer is considered a part of the trusted environment on the side of the process. The NN-MPC controller is evaluated in a separate server-side Python script, simulating a cloud environment. The encryption/decryption layer and NN-MPC communicate via HTTP. To implement operations over encrypted data, we use Python library *TenSEAL*[6] [15] based on *Microsoft SEAL* [16]. The general algorithm of closed-loop control over encrypted data is shown in Alg. 1. Firstly, the data owner (process) generates cryptographic keys. These are public key pk, secret key sk, Galois key gk, and re-linearization key rk. During the key exchange, pk, gk, rk are sent to the controller and later used for homomorphic evaluation. During the control,

[6]https://github.com/OpenMined/TenSEAL

Algorithm 1: Implementation of HE control

```
1  Key generation (process side - data owner):
2      generate: $pk, sk, gk, rk$
3      keep: $sk$
4  Key exchange:
5      send: $pk, gk, rk \rightarrow$ Controller
6  Closed-loop:
7      Process (process side - data owner)
8          Measure states $\rightarrow x_k$
9          Encode states $\rightarrow x'_k = \mathcal{E}(x_k, \Delta)$
10         Encrypt states $\rightarrow x^E_k = E(x'_k, sk)$
11     NN-MPC Controller (cloud environmet)
12         Evaluate $\rightarrow u^E_k = \text{NN}(x^E_k, pk, gk, rk)$
13     Process (process side - data owner)
14         Decrypt $\rightarrow u'_k = D(u^E_k, sk)$
15         Decode $\rightarrow u_k = \mathcal{D}(u'_k, \Delta)$
16         Apply $\rightarrow u_k$
```

a vector of state measurements is encoded with the scale Δ into plaintext and encrypted using sk. Encrypted states x^E_k are then sent to the NN-MPC controller to be evaluated. Public key pk is used in every homomorphic operation (additions, multiplications), rk is required in multiplication between two ciphertexts (evaluation of activation functions), and gk is used for matrix multiplications. The resulting encrypted control u^E_k is then sent back to the process, decrypted using sk, decoded, and applied to the process.

The biochemical reactor was controlled on a simulated timespan of 150 hours with a control sampling period of 30 minutes for original MPC, NN-MPC, and NN-MPC over encrypted data. To illustrate the disturbance rejection performance of the controllers, we initiated control in an arbitrary combination of states $x_\text{init} = [0.81, 11.82, 31.50, 6.15, 28.06, 25.12]$. Afterward, two additional impulse disturbances on process input (coolant flow) were applied in time 100h ($F_\text{c} = 70$) and 125h ($F_\text{c} = 0$). The results are shown in figure 4. Both the original MPC (blue line) and NN-MPC (orange line) were able to compensate for disturbances and bring the process states to the desired operating point. MPC provides a faster response and shorter settling time than NN-MPC. As a result of a rather large initial state disturbance, the NN-MPC tent to slightly violate the bottom input constraint. However, this violation did not apply since the inputs are trimmed before being sent to the process. The NN-MPC control over encrypted data (black line) is almost identical to plain NN-MPC, with minor numerical discrepancy caused by cryptographic noise and rounding errors introduced during message encoding.

G. Limitations

While the primary goal of HE-based process control (the ability to compute over secured data) was achieved, several

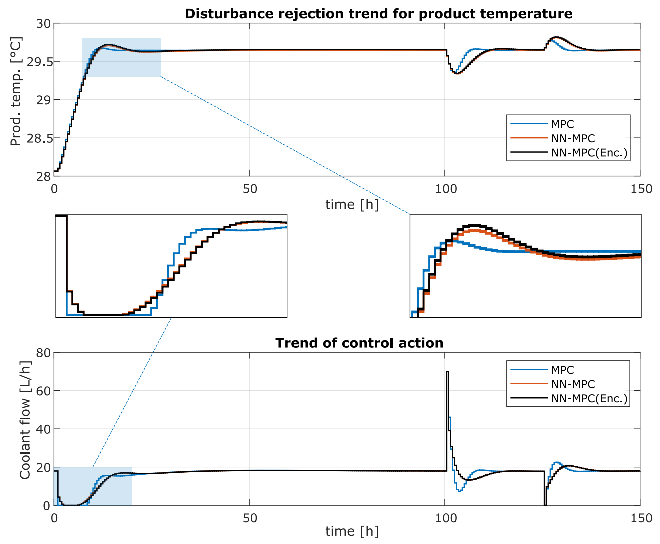

Fig. 4. Disturbance rejection control trends for product temperature. The blue line shows the control trajectory of the original MPC, the orange line shows the performance of the NN-MPC controller, and the black line shows the NN-MPC controller over encrypted data.

limitations must be considered. The RLWE-based cryptosystems tend to be computationally demanding. In the presented control scenario, a single HE inference of the NN-MPC controller took on average 10.11 seconds (AMD Ryzen 3950X CPU, 128GB DRR4 RAM). The per-layer computational time grows quadratically with the number of neurons (size of weight matrices). While the complexity grows only linearly with an increasing number of layers, more than three hidden layers would require higher multiplicative depth, thus bigger polynomial modulus degree N. This would lead to an impractical cryptosystem setup due to the computational demand and size of the keys. For the CKKS setup used in this work (Table II), the size of the public key was 1.97MB, the secret key 1.01MB, the re-linearization key 21.69MB, and Galois keys 564.81MB.

IV. Conclusions

This paper shows that even complex controllers like MPC can be implemented in an approximated form on HE frameworks if the implementer is willing to sacrifice some of the original control performance and numerical precision. The main benefit, i.e., the preservation of data privacy, comes with the cost of increased computational and memory demand that can vary based on the setup of the cryptosystem. The practicality of such a setup depends on a specific application. The homomorphically evaluated NN-MPC of the same structure as the one presented in this paper would not be implementable for controlling processes with fast dynamics. However, the presented approach is viable for slow processes like the biochemical reactor. One of the options for reducing both the computational time and size of the transferred data would be decreasing the multiplicative depth of NN. This can be done either by decreasing the number of layers and/or using a square function as activation. This would allow for N to be 4096 or even 2048, reducing the computational overhead significantly at the cost of creating a bigger discrepancy between MPC and NN-MPC. The computational time can also be reduced by decreasing the number of neurons in layers, resulting in homomorphic multiplication between smaller matrices and vectors.

Acknowledgment

This research is funded by the Slovak Research and Development Agency under the project APVV-21-0019, by the Scientific Grant Agency of the Slovak Republic under the grant VEGA 1/0691/21, and by the European Commission under the grant no. 101079342 (Fostering Opportunities Towards Slovak Excellence in Advanced Control for Smart Industries).

References

[1] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," vol. 51, jul 2018.

[2] 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.

[3] 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.

[4] J. Tran, F. Farokhi, M. Cantoni, and I. Shames, "Implementing homomorphic encryption based secure feedback control," *Control Engineering Practice*, vol. 97, p. 104350, 04 2020.

[5] Y. Lin, F. Farokhi, I. Shames, and D. Nešić, "Secure control of nonlinear systems using semi-homomorphic encryption," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 5002–5007, 2018.

[6] A. B. Alexandru, M. Morari, and G. J. Pappas, "Cloud-based mpc with encrypted data," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 5014–5019, 2018.

[7] 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)*, pp. 5469–5476, 2020.

[8] O. Regev, "The learning with errors problem (invited survey)," in *2010 IEEE 25th Annual Conference on Computational Complexity*, pp. 191–204, 2010.

[9] 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)*, pp. 4907–4914, 2021.

[10] 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.), pp. 409–437, Springer International Publishing, 2017.

[11] W. Jung, E. Lee, S. Kim, J. Kim, N. Kim, K. Lee, C. Min, J. H. Cheon, and J. H. Ahn, "Accelerating fully homomorphic encryption through architecture-centric analysis and optimization," *IEEE Access*, vol. 9, pp. 98772–98789, 2021.

[12] K. Y. F. Lip, E. García-Ríos, C. E. Costa, J. M. Guillamón, L. Domingues, J. Teixeira, and W. M. van Gulik, "Selection and subsequent physiological characterization of industrial saccharomyces cerevisiae strains during continuous growth at sub-and-supra optimal temperatures," *Biotechnology Reports*, vol. 26, p. e00462, 2020.

[13] M. Ławryńczuk, "Modelling and nonlinear predictive control of a yeast fermentation biochemical reactor using neural networks," *Chemical Engineering Journal*, vol. 145, no. 2, pp. 290–307, 2008.

[14] S. Obla, X. Gong, A. Aloufi, P. Hu, and D. Takabi, "Effective activation functions for homomorphic evaluation of deep neural networks," *IEEE Access*, vol. 8, pp. 153098–153112, 2020.

[15] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "Tenseal: A library for encrypted tensor operations using homomorphic encryption," 2021.

[16] "Microsoft SEAL (release 4.1)." https://github.com/Microsoft/SEAL, Jan. 2023. Microsoft Research, Redmond, WA.