IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received 14 September 2023, accepted 26 September 2023, date of publication 4 October 2023,
date of current version 11 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3322035

METHODS

Guidelines for Secure Process Control: Harnessing the Power of Homomorphic Encryption and State Feedback Control

M. FURKA, M. KALÚZ, M. FIKAR, AND M. KLAUČO, (Member, IEEE)
Faculty of Chemical and Food Technology, Slovak University of Technology in Bratislava, 81237 Bratislava, Slovakia

Corresponding author: M. Furka (matus.furka@stuba.sk)

This work was supported in part by the Scientific Grant Agency of the Slovak Republic under Grant 1/0545/20, in part by the Slovak Research and Development Agency under Project APVV-20-0261 and Project APVV-21-0019, and in part by the European Union's Horizon Europe through the Fostering Opportunities Toward Slovak Excellence in Advanced Control for Smart Industries under Grant 101079342.

ABSTRACT The paper presents applications of homomorphic cryptographic schemes in process control. Homomorphic encryption is a particular type of encryption that allows performing mathematical operations over encrypted data. Therefore the user can use third-party services as controllers without revealing any vulnerable information, like process data. Homomorphic properties make it possible to perform data aggregation or evaluate control actions without having any knowledge about the process data. Our paper explores two fully homomorphic cryptographic schemes, namely Brakerski/Fan-Vercauteren (BFV) and Cheon-Kim-Kim-Song (CKKS) cryptosystems. Each scheme is briefly described with its advantages and drawbacks, conditions, and applications. We present a case study involving linear-quadratic (LQ) control strategy implemented in an encrypted setup. This serves for comparison and a step-by-step guide for implementing encrypted process control.

INDEX TERMS Modern cryptography tools, homomorphic encryption, state feedback control.

I. INTRODUCTION

The modern industry heavily relies on information sharing and network control systems. Hence, the need for data security and privacy became increasingly important aspects.

The traditional encryption schemes such as AES [1] or RSA [2] are dominant in securing the world's network traffic, most of computer systems, and software applications. Homomorphic encryption (HE) schemes are designed to allow certain types of operations over encrypted data without the need for prior decryption. Some of the early HE schemes, commonly known as partially homomorphic encryption (PHE) schemes allow either an unbounded number of modular multiplications over the ciphertexts (RSA [2], ElGamal [3]) or an unbounded number of modular additions (Benaloh [4], Paillier [5]).

Fully homomorphic encryption (FHE) first introduced by Gentry [6] is an encryption technique that allows one to evaluate arbitrary functions over the encrypted data without decryption. Several cryptographic systems and frameworks have been introduced in recent years that provide extended FHE properties. The notable ones are BFV [7], BGV [8], THFE [9], FHEW [10], and HEAAN/CKKS [11].

Homomorphic encryption finds its use in a variety of applications, mainly in those where the preservation of privacy is of imminent interest. Many applications of HE have been proposed or directly implemented in recent years. These tend to secure privacy in various fields such as transportation [12], geo-location [13], e-voting [14], to deal with an analysis of a sensitive medical [15] or biometric data [16], [17]. The utilization of HE came to a point when it is considered to become a mandatory standard for certain applications [18].

Several HE-related studies have been conducted in the field of control systems as show in [19], [20], [21], and [22].

The associate editor coordinating the review of this manuscript and approving it for publication was S. K. Hafizul Islam.

However, it is fair to say that HE-related control schemes are mostly in the proof-of-concept state, and we will have to wait for their adoption in industrial applications. Many of these studies rely on the utilization of Paillier cryptosystem (PCS), despite the fact that it does not support fully homomorphic multiplication. Nevertheless, PCS provides sufficient security, performance, and homomorphic framework to support applications such as: linear controllers on dedicated hardware [23]; nonlinear [24] and distributed [25] network control systems; cooperative schemes [26]; quadratic optimization [27]; implicit [28] MPC; and explicit MPC [29] respectively. However, modern HE schemes have usually, several cryptographic parameters, which often depend on each other, thus the setup of the HE scheme is not trivial.

The novelties of this paper are:

- comprehensive guideline of the configuration of parameters of homomorphic cryptographic schemes leading to satisfactory implementation in closed-loop control applications,
- exhaustive analysis of encoding, encryption, and pre-processing procedures of BFV and CKKS frameworks with respect to varying parameters of cryptographic configurations.

We emphasize, that selecting encoding and encryption parameters in BFV and CKKS is crucial to balance security, performance, and numerical precision. The cryptographic parameters of both schemes are highly interdependent, thus the selection of each parameter is a non-trivial task. Hence, we provide a step-by-step algorithm, that results in proper cryptographic configuration regarding the process control requirements.

The paper is organized as follows: In Section III, we discuss general terms of homomorphic encryption and its practical usage in various applications. Next, in Section IV we present BFV and CKKS cryptographic schemes, along with their homomorphic properties. Section V is dedicated to the description of cryptographic parameters in terms of practical implementation and their impact on encoding, encryption, and numerical correctness. Section V provides the guideline for cryptographic parameter setup for each scheme according to application requirements. In Section VI, we present simulation results of encrypted closed-loop state feedback control, while in Section VII we conclude the paper with evaluation of achieved results.

II. RELATED WORK

In recent years, several open-source libraries have been created considering different HE schemes for various applications. However, they generally provide only a few examples with correct cryptographic parameters, but not the guideline itself. The setting of cryptographic parameters is a non-trivial the task, especially for LWE-based encryption frameworks. Moreover, the cryptosystem configuration in these libraries is not automated and has to be done manually. So far, only a few proposals have looked at the construction of guidelines for fully homomorphic encryption frameworks [30], [31], [32]. Authors in [30] provide a guideline for BGV [33] cryptosystem implemented with HElib library [34], while in [31] the authors discuss and construct instructions for CKKS [11] configuration with Lattigo library [35]. In [32], the author focuses on BFV [7] guideline considering its implementation within Microsoft SEAL [36]. Our proposal provides a guideline involving simple computational steps resulting in the correct setting of cryptographic parameters for both BFV and CKKS while utilizing their implementation in *TenSEAL* library. The configuration guideline leads the non-specialists towards correct cryptographic parameter setup for secured process control applications. Furthermore, the guideline respects the required measurement and controller gain precision, while ensuring the security level at least 128 bits.

III. HOMOMORPHIC ENCRYPTION

A. GENERAL & PRACTICAL ASPECTS

Homomorphic encryption (HE) algorithms allow to perform mathematical operations on encrypted data without the need for its decryption. This brings several benefits since we can utilize third-party services without revealing data.

Before we provide a basic description of cryptographic schemes, we define three levels of encrypted information as follows:

- *message* commonly noted as m and is defined as a "raw" scalar or a vector (positive or negative integer or floating point numbers) that needs to be processed before it can be encrypted. Raw numbers are processed via encoding procedures resulting in the form of messages suitable for cryptographic schemes since their algorithms are based on modular (integer) arithmetic.
- *plaintext* is the form of message, which is acceptable for cryptographic schemes. Plaintext is usually noted as p and has a form of positive integer or polynomial with integer coefficients depending on the cryptographic scheme. After decryption, plaintexts are decoded to recover the original value of message m or the result from homomorphic operation over ciphertexts.
- *ciphertext* often marked as c represents the encrypted plaintext p hiding the value of the original message in large integers or polynomials.

These definitions describe the main differences between messages, plaintexts and ciphertexts. For example, for Paillier cryptosystem may the values of m, p and c have the form

$$\underbrace{-0.5917}_{m} \overset{f_{\mathrm{D}}}{\underset{f_{\mathrm{E}}}{\leftrightarrows}} \underbrace{4177750720}_{p} \overset{\mathcal{D}}{\underset{\mathcal{E}}{\leftrightarrows}} \underbrace{741304627931510405}_{c} \tag{1}$$

where f_{E} denotes encoding procedure, f_{D} is decoding, $\mathcal{E}$ stands for encryption and $\mathcal{D}$ for decryption. The rest of further used notation is listed in Table 1.

TABLE 1. Notation.

Symbol	Description	Symbol	Description	Symbol	Description
m	message	$\mathbb{Z}_n^+$	set of positive integers less than n	k_{eval}	evaluation key
p	plaintext	$\mathbb{R}$	set of real numbers	k_{gal}	galois key
c	ciphertext	$\mathbb{C}$	set of complex numbers	$\mathcal{R}$	polynomial ring $\mathbb{Z}[X]/(X^n+1)$ with degree n
$\mathcal{N}$	modulus degree	$f_{\text{E}}(\cdot)$	encoding function	$\mathcal{R}_n$	set of polynomials in $\mathcal{R}$ with coefficients in $\mathbb{Z}_n^+$
q	coefficient modulus	$f_{\text{D}}(\cdot)$	decoding function	$\lfloor\cdot\rceil$	round to nearest integer
τ	plaintext modulus	$\mathcal{E}(\cdot)$	encryption function	$\lfloor\cdot\rfloor$	round to nearest smaller integer
Δ	scaling factor	$\mathcal{D}(\cdot)$	decryption function	$[a]_n$	$a \mod n$
π	mapping function	k_{pub}	public key	Θ	uniform distribution over integers
$\mathbb{Z}$	set of integers	k_{pvt}	private (secure) key	Φ	discrete Gaussian distribution over integers

The HE algorithms are generally divided into two main groups, partially (PHE) and fully (FHE) homomorphic encryption algorithms. This classification is obtained regarding the possibility of providing either homomorphic addition or homomorphic multiplication (PHE) or both (FHE). The choice of a cryptosystem is generally specified given the application or user requirements.

In the following lines, we present some of the many applications where homomorphic algorithms find their use, such as:

- *Voting* – One of the simple examples of using HE schemes is the voting system. It is usually a matter of counting individual votes, which are anonymous. For this purpose, cryptographic schemes providing homomorphic addition find great use [37].
- *Data processing* – Nowadays exist a lot of cloud applications that can process data using their algorithms and can return results in various forms. However, sharing personal or vulnerable data in public space is not a good option. Here, HE algorithms can find their use. The cloud application works with ciphertexts but is still able to process the real data thanks to homomorphism. The outcome remains encrypted and only the data owner can decrypt the result [38].
- *Process Control* – Based on the process or requirements, the partially or fully HE algorithms can be used. There are three types of scenarios for including encryption schemes into process control algorithms. One aim can be to secure only the data and the controller remains public. On the other hand, sometimes the control algorithms are more valuable than the data itself. Finally, the fully encrypted closed-loop control can be employed, where both the data and the controller are encrypted. The type of HE scheme is chosen for the given control scenario [29], [39].

In this paper, we focus on the third scenario, the process control secured via fully homomorphic encryption frameworks. The majority of current research is focused on secured data processing, not process control applications. Moreover, papers oriented towards secured process control consider partially homomorphic encryption frameworks. The main drawback of PHE schemes relies on securing only one element of a closed-loop scheme, thus either the process measurements or controller parameters remain unencrypted. By involving fully homomorphic frameworks, we provide cryptographic security for both, process data and controller.

B. HOMOMORPHIC ENCRYPTION IN PROCESS CONTROL

We consider homomorphic encryption algorithms to secure the data flow between multiple layers of control systems, like between the sensor and the controller, or between the controller and the actuator. Such a feature is heavily needed in the process industry, where sensors, controllers, and actuators are rarely tightly coupled in one device but can be connected via an intranet, wireless networks or they are geographically separated.

We operate with homomorphic encryption procedures using the asymmetric key setup. Hence, all messages are encrypted with a public key but can be decrypted only with a private key. In process control, we consider the process as the holder of both keys, whereas in abstract terms, the sensor encrypts process measurements with the public key, and sends them to the controller, which has access only to the public key. The process finally receives encrypted values of manipulated variables, which are decrypted on the process side with the private key.

Since the controller has no access to decrypted values of process measurements, we exploit homomorphic properties of several cryptographic schemes allowing us to perform mathematical operations over encrypted values. Note, that the public key is distributed alongside the encrypted measured values since the mathematical operations (like addition or multiplication) cannot be performed without the knowledge of the public key. Recall, that the encrypted values cannot be decrypted without the knowledge of the private key, which remains stored only on the process side.

This paper exploits the possibilities of implementing traditional control algorithms in a cryptographically secure framework. We offer step-by-step procedures of how the implementation of LQ controllers is done within two FHE schemes.

IV. HOMOMORPHIC ENCRYPTION ALGORITHMS

This section provides the implementation details of incorporating a fully homomorphic framework in closed-loop control. First, we introduce the specifics of the BFV

(Section IV-A) and CKKS cryptosystems (Section IV-B). Second, we introduce the guidelines for setting up individual parameters of BFV and CKKS for this class of applications.

Given the several existing encryption frameworks providing cryptographical security for various applications, we emphasize the most important reasons directing our choice towards the use of BFV and CKKS cryptosystems:

1) *Future-proof Security*: Both CKKS and BFV are based on hardness assumption known as Ring Learning With Errors (RLWE) and are also classified as lattice-based or post-quantum cryptosystems. Today, there are no known algorithms that would solve the RLWE in sub-exponential time complexity. Contrary to RLWE, other cryptosystems that rely on large integer factorization (IF) or hardness of discrete logarithm problem (DLP) are proved to be vulnerable to Shor's quantum algorithm. Some of the most popular homomorphic cryptosystems belong to this category: RSA (IF), Paillier cryptosystem (IF), Benaloh cryptosystem (IF), and ElGamal (DLP).
2) *Homomorphic Features*: Both BFV and CKKS can perform homomorphic addition and multiplication while respecting the predefined depth of the arithmetic circuit. Encoding and decoding procedures are implemented in libraries by default.
3) *Process Control Features*: Encryption libraries provide additional features built on top of cryptosystems. In the case of Microsoft SEAL, we can utilize efficient algorithms for polynomial evaluation, matrix/vector addition, matrix-vector multiplication, and vector rotation. These operations are often used in control algorithms.
4) *Parallelism*: CKKS and BFV operate in a Single Instruction/Multiple Data (SIMD) fashion. Multiple numerical messages can be encoded into the slots of a single ciphertext, and the subsequent operations are carried out over all the slots. The implementer can utilize quite a large number of $\mathcal{N}/2$ (CKKS) or $\mathcal{N}$ (BFV) slots with $\mathcal{N}$ being the polynomial modulus degree (e.g., $\mathcal{N}$: 1024, 2048, 4096, 8192, etc.). This brings significant capabilities for parallel data processing.

In the following lines we describe cryptographic schemes and homomorphic properties of both BFV and CKKS encryption frameworks.

A. BRAKERSKI/FAN-VERCAUTEREN (BFV) CRYPTOSYSTEM

The *BFV* [7] scheme is a leveled HE scheme, which allows performing the modular arithmetic on encrypted integers. The number of levels is defined as the number of possible multiplications over a single ciphertext in one arithmetic circuit. BFV operates with vector messages encoded and encrypted into polynomials. The authors relied on the Brakerski-Gentry-Vaikuntanathan (BGV) [8] scheme, where they changed the *Learning With Errors* (LWE) setting to *Ring Learning With Errors* (RLWE). Given the RLWE setup, the ciphertexts in BFV are represented by tuples of two polynomials, i.e., $c = (c[0], c[1])$, thus having the size of 2. The ciphertext in BFV is a space bounded by integer modulus q including message payload and cryptographic noise. These two elements have to be separated and must not interact during homomorphic operations, otherwise, the cryptographic noise will spoil the original message or result. The payload is limited by interval $(0, \tau - 1)$ with τ being a plaintext modulus and cryptographic noise can move in the interval $(\tau, q - 1)$. The decryption works correctly while both message payload and cryptographic noise remain in predefined intervals.

The BFV scheme provides the exact computations, therefore, it is the best choice when requiring exact results.

1) HOMOMORPHIC SCHEME

The BFV scheme provides the following procedures:

- **Key generation**: Given security parameter λ, the private key k_{pvt} is sampled from uniform distribution over integers Θ. By using this private key and integer coefficient modulus q, the generation algorithm computes the public key in the form $k_{\text{pub}} = ([-Xk_{\text{pvt}} + e]_q, X)$ with X sampled from Θ and e sampled from discrete Gaussian distribution over integers Φ. Also, two important keys are generated:
  - evaluation key k_{eval} used to relinearize the ciphertext obtained from multiplication back to size 2.
  - galois key k_{gal} allows to rotate the vectors of encrypted values to effectively perform homomorphic multiplications.
- **Encoding**: The message vector $\mathbf{m}$ in BFV scheme is encoded into plaintext p having the form of polynomial whose space is a ring $R_\tau = \mathbb{Z}_\tau^+[X]/(X^{\mathcal{N}} + 1)$, where τ is integer plaintext modulus and $\mathcal{N}$ is polynomial modulus degree. The encoding function $f_{\text{E}}(\cdot)$ takes as input plaintext modulus τ and message vector $\mathbf{m}$

$$p = f_{\text{E}}(\mathbf{m}, \tau), \qquad \mathbf{m} = [m_1, m_2, \ldots, m_{\mathcal{N}}] \in \mathbb{Z}^{\mathcal{N}} \quad (2)$$

and returns plaintext polynomial p in form

$$p = [p_0]_\tau X^0 + [p_1]_\tau X^1 + \cdots + [p_{\mathcal{N}-1}]_\tau X^{\mathcal{N}-1} \quad (3)$$

where $p_0, p_1, \ldots, p_{N-1}$ are positive integer coefficients smaller than τ.
- **Encryption**: The polynomial space for ciphertexts is defined as $\mathcal{R}_q^2 = \mathbb{Z}_q^+[X]/(X^{\mathcal{N}} + 1)$, where 2 in upper subscript stands for tuple of two elements. To encrypt the plaintext p, the public key k_{pub} and three random polynomials v_0, v_1, v_2 with coefficients sampled from Φ are used to get the ciphertext c defined as tuple

$$c = (c[0], c[1]), \quad (4)$$

with two coordinates $c[0]$ and $c[1]$ computed as

$$c[0] = [\delta \cdot p + k_{\text{pub}}[0] \cdot v_0 + v_1]_q, \quad (5a)$$

$$c[1] = [k_{\mathrm{pub}}[1] \cdot v_0 + v_2]_q. \tag{5b}$$

where $\delta = \lfloor q/\tau \rfloor$ and q is integer coefficient modulus fulfilling $1 < \tau < q$.

- **Decryption**: To decrypt ciphertext c, the decryption function takes as input private key k_{pvt} and solves

$$p = \left[\left\lfloor \frac{\tau \cdot [c[0] + c[1] \cdot k_{\mathrm{pvt}}]_q}{q} \right\rceil\right]_\tau . \tag{6}$$

- **Decoding**: The original message or resulting vector from homomorphic operation is recovered from c by decoding the plaintext p

$$\mathbf{m} = f_{\mathrm{D}}(p, \tau) \tag{7}$$

where $f_{\mathrm{D}}(\cdot)$ denotes decoding function.

2) HOMOMORPHIC PROPERTIES

The BFV scheme allows us to perform both homomorphic addition and homomorphic multiplication operations with ciphertexts. Thus the cryptographic properties of BFV can be summarized as follows:

- *Ciphertext Addition*: To add messages $\mathbf{m_1}$ and $\mathbf{m_2}$ homomorphically, first step is to encode and encrypt them into ciphertexts c_1 and c_2 as

$$c_1 = \mathcal{E}(f_{\mathrm{E}}(\mathbf{m_1})) = (c_1[0], c_1[1]), \tag{8a}$$
$$c_2 = \mathcal{E}(f_{\mathrm{E}}(\mathbf{m_2})) = (c_2[0], c_2[1]), \tag{8b}$$

and then compute

$$c_{\mathrm{A}} = c_1 + c_2 = (c_{\mathrm{A}}[0], c_{\mathrm{A}}[1]), \tag{9a}$$
$$c_{\mathrm{A}}[0] = [c_1[0] + c_2[0]]_q, \tag{9b}$$
$$c_{\mathrm{A}}[1] = [c_1[1] + c_2[1]]_q, \tag{9c}$$

which is the encryption of $\mathbf{m_1} + \mathbf{m_2}$.

- *Plaintext Multiplication*: The BFV scheme provides multiplication of ciphertext by plaintext (constant) defined by relation

$$c_{\mathrm{P}} = p \cdot c_1 = ([p \cdot c_{\mathrm{P}}[0]]_q, [p \cdot c_{\mathrm{P}}[1]]_q), \tag{10}$$

where p is plaintext or public vector.

- *Ciphertext Multiplication*: Multiplication of two messages $\mathbf{m_1}$ and $\mathbf{m_2}$ encoded and encrypted as c_1 and c_2 is defined as

$$c_{\mathrm{M}} = c_1 \cdot c_2 = (c_{\mathrm{M}}[0], c_{\mathrm{M}}[1], c_{\mathrm{M}}[2]), \tag{11a}$$
$$c_{\mathrm{M}}[0] = \left[\left\lfloor \frac{\tau \cdot c_1[0] \cdot c_2[0]}{q} \right\rceil\right]_q , \tag{11b}$$
$$c_{\mathrm{M}}[1] = \left[\left\lfloor \frac{\tau \cdot (c_1[0] \cdot c_2[1] + c_2[0] \cdot c_1[1])}{q} \right\rceil\right]_q , \tag{11c}$$
$$c_{\mathrm{M}}[2] = \left[\left\lfloor \frac{\tau \cdot c_1[1] \cdot c_2[1]}{q} \right\rceil\right]_q . \tag{11d}$$

To take the size of the ciphertext c_{M} back to 2, the relinearization procedure $f_{\mathrm{R}}(\cdot)$ using the evaluation key k_{eval} is carried out in the form

$$c_{\mathrm{R}} = f_{\mathrm{R}}(c_{\mathrm{M}}, k_{\mathrm{eval}}) = (c_{\mathrm{R}}[0], c_{\mathrm{R}}[1]), \tag{12a}$$
$$c_{\mathrm{R}}[0] = c_{\mathrm{M}}[0] + \sum_{i=0}^{k} k_{\mathrm{eval}}[i][0] \cdot c_{\mathrm{M}}[2]^{(i)}, \tag{12b}$$
$$c_{\mathrm{R}}[1] = c_{\mathrm{M}}[1] + \sum_{i=0}^{k} k_{\mathrm{eval}}[i][1] \cdot c_{\mathrm{M}}[2]^{(i)}, \tag{12c}$$

where $c_{\mathrm{R}}[0]$ and $c_{\mathrm{R}}[1]$ are coordinates of final relinearized ciphertext c_{R}.

B. CHEON-KIM-KIM-SONG (CKKS) CRYPTOSYSTEM

The CKKS [11] is a leveled HE scheme that provides arithmetic operations with real and complex numbers, and yields approximated results. Similarly to BFV, the operations in CKKS are carried out over the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q^+[X]/f(X)$, where q is coefficient modulus and $f(X) = X^{\mathcal{N}} + 1$ is a polynomial known as polynomial modulus, while $\mathcal{N}$ is chosen to be a power of two. The elements of $\mathcal{R}_q$ are polynomials with integer coefficients smaller than q and of degree at most $(\mathcal{N} - 1)$. The CKKS uses these polynomials as elements of both plaintexts and ciphertexts. The main difference in the ciphertext structure between BFV and CKKS is that CKKS adds cryptographic noise to the underlying ciphertext payload. To avoid the loss of numerical correctness, before the encryption, the payload is multiplied by a scaling factor Δ that moves the significant bits of the message to the left. After that, the noise is added to less significant bits of the message space. By encrypting two numerical vector messages $\mathbf{m_1}$ and $\mathbf{m_2}$, the underlying ciphertext payloads would be $\mathbf{m_1} \cdot \Delta$ and $\mathbf{m_2} \cdot \Delta$. It is easy to see that result of ciphertext multiplication $(\mathbf{m_1} \times \mathbf{m_2}) \cdot \Delta^2$ grows by a factor of Δ. To avoid uncontrolled growth of payload inside the message space, the authors of CKKS implement a technique called *rescaling*. During this procedure, the resulting ciphertext is reduced by Δ, which not only brings the message to the original scale but also resets the magnitude of cryptographic noise in the payload. The downside is that the coefficient modulus of ciphertext polynomials is also rescaled to q/Δ. To cope with this issue, the authors introduced a concept of levels. The fresh ciphertext uses coefficient modulus q at the highest level L that represents the maximum number of consecutive multiplications. After every multiplication, the modulus is reduced by Δ. The ciphertext modulus at the highest level $q = q_0 \cdot q_1 \cdot \ldots \cdot q_L$ consists of distinct primes, defined by their bit sizes, where special prime q_0 is the base modulus, and the number of primes $q_1, \cdots, q_L$ defines the maximum multiplicative depth of the arithmetic circuit. Additionally, the public key, relinearization key, and rotation key are defined at level $L + 1$ and use another special prime q_{L+1}. The bit size of these primes is defined during the setup of the CKKS scheme in the form of an array (sec. V-A, eq. 24).

1) HOMOMORPHIC SCHEME

The functionality of the CKKS scheme is based on the following procedures:

- **Key generation**: Secret key k_{pvt} is an $\mathcal{N}$-degree polynomial sampled from $\mathcal{R}_2$ with coefficient in $\{-1, 0, 1\}$ chosen from distribution described in [11, Sec. 3.4]. The public key is then generated as a tuple of two polynomials $k_{\text{pub}} = (k_{\text{pub}}[0], k_{\text{pub}}[1]) = ([-a \cdot k_{\text{pvt}} + e]_q, a)$, where a is polynomial uniformly sampled from $\mathcal{R}_q$, and e is a random error polynomial with coefficients sampled from discrete Gaussian distribution Φ, defined by *Homomorphic Encryption Security Standard* [40]. Similar to BFV, the CKKS also works with:
  - evaluation key k_{eval} used to reduce the number of ciphertext elements (polynomials) from three back to two after every homomorphic multiplication,
  - Galois key k_{gal} used to perform encrypted vector rotation on ciphertexts.
- **Encoding**: During this procedure, the message payload in the form of a complex vector
$$\mathbf{m} = [m_1, \ldots, m_{\mathcal{N}/2}] \in \mathbb{C}^{\mathcal{N}/2} \tag{13}$$
is encoded into plaintext polynomial $p \in \mathcal{R}_q$. Both encoding and decoding are done via mapping function $\pi(\cdot)$ that performs complex canonical embedding (see. [11, Sec. 2.2]). The transformation of message $\mathbf{m}$ into plaintext p is then performed via encoding function $f_{\text{E}}(\cdot)$ such that
$$p = f_{\text{E}}(\mathbf{m}, \Delta) = \left\lfloor \Delta \cdot \pi^{-1}(\mathbf{m}) \right\rceil . \tag{14}$$
- **Encryption**: The ciphertext $c = (c[0], c[1])$, formed by a tuple of two polynomials, is obtained by the encryption function
$$c = \mathcal{E}(p, k_{\text{pub}}), \tag{15}$$
calculated element-wise as
$$c[0] = [k_{\text{pub}}[0] \cdot u + e_0 + p]_q, \tag{16a}$$
$$c[1] = [k_{\text{pub}}[1] \cdot u + e_1]_q, \tag{16b}$$
where $u \in \mathcal{R}_2$ is a random polynomial with signed binary coefficients, and $e_0, e_1 \in \Phi$ are random error polynomials.
- **Decryption**: The decryption of ciphertext c on the private key k_{pvt} is done by evaluating
$$\tilde{p} = \mathcal{D}(c, k_{\text{pvt}}) = [c[0] + c[1] \cdot k_{\text{pvt}}]_q = p + e. \tag{17}$$
It is obvious, that the decryption $\tilde{p}$ is not exactly p, but $p+e$. If the noise e is small enough, the original plaintext message p and decrypted $\tilde{p}$ should be very close.
- **Decoding**: When decoding, the decrypted plaintext is divided by the scale and mapped back to $\mathbb{C}^{\mathcal{N}/2}$ using a decoding function
$$\tilde{\mathbf{m}} = f_{\text{D}}(\tilde{p}, \Delta) = \pi\left(\frac{1}{\Delta} \cdot \tilde{p}\right). \tag{18}$$

2) HOMOMORPHIC PROPERTIES

The CKKS scheme provides the following homomorphic properties:

- *Ciphertext Addition*: Assume we have two vector messages $\mathbf{m_1}$ and $\mathbf{m_2}$ encoded and encrypted as c_1 and c_2, where
$$c_1 = \mathcal{E}(f_{\text{E}}(\mathbf{m_1})) = (c_1[0], c_1[1]), \tag{19a}$$
$$c_2 = \mathcal{E}(f_{\text{E}}(\mathbf{m_2})) = (c_2[0], c_2[1]). \tag{19b}$$
The homomorphic addition is computed as
$$c_{\text{A}} = c_1 + c_2 = (c_{\text{A}}[0], c_{\text{A}}[1]) \tag{20a}$$
$$c_{\text{A}}[0] = [c_1[0] + c_2[0]]_q, \tag{20b}$$
$$c_{\text{A}}[1] = [c_1[1] + c_2[1]]_q, \tag{20c}$$
which after decryption and decoding results in $\mathbf{m_1} + \mathbf{m_2} + \mathbf{e} \approx \mathbf{m_1} + \mathbf{m_2}$.
- *Plaintext Multiplication*: Multiplication of ciphertext c by plaintext p is computationally simpler and is performed as
$$c_{\text{P}} = p \cdot c = ([p \cdot c[0]]_q, [p \cdot c[1]]_q). \tag{21}$$
- *Ciphertext Multiplication*: The CKKS scheme also provides homomorphic multiplication of two ciphertexts c_1 and c_2 defined as
$$c_{\text{M}} = c_1 \cdot c_2 = (c_{\text{M}}[0], c_{\text{M}}[1], c_{\text{M}}[2]), \tag{22a}$$
$$c_{\text{M}}[0] = [c_1[0] \cdot c_2[1]]_q\,, \tag{22b}$$
$$c_{\text{M}}[1] = [c_1[0] \cdot c_2[1] + c_2[0] \cdot c_1[1]]_q\,, \tag{22c}$$
$$c_{\text{M}}[2] = [c_1[1] \cdot c_2[1]]_q\,. \tag{22d}$$
Since the size of the ciphertext grows, the resulting ciphertext c_{M} requires the relinearization procedure $f_{\text{R}}(\cdot)$ in order to ensure that the ciphertext has 2 elements. Ciphertext c_{M} is relinearized as
$$c_{\text{R}} = f_{\text{R}}(c_{\text{M}}, k_{\text{eval}}) = (c_{\text{R}}[0], c_{\text{R}}[1]) \tag{23a}$$
$$c_{\text{R}}[0] = c_{\text{M}}[0] + [\lfloor \gamma^{-1} \cdot c_{\text{M}}[2] \cdot k_{\text{eval}} \rceil]_q \tag{23b}$$
$$c_{\text{R}}[1] = c_{\text{M}}[1] + [\lfloor \gamma^{-1} \cdot c_{\text{M}}[2] \cdot k_{\text{eval}} \rceil]_q \tag{23c}$$
where $f_{\text{R}}(\cdot)$ is relinearization function, c_{R} is relinearized ciphertext, γ is a big integer and k_{eval} is evaluation key described in [11].

V. EXPERIMENTAL SETUP

In our case, the simulation experiments were performed using *Python* language. All the results presented in this paper were obtained on a computer with 3.4 GHz processor, 128 GB of RAM and 64-bit operating system. In order to use above mentioned cryptographic schemes we use the *TenSEAL* python library [41] built on the *Microsoft SEAL* [36]. This section presents in detail, the effects and importance of individual parameters for successful implementation of homomorphic encryption in process control.

A. CRYPTOGRAPHIC PARAMETERS

Library *TenSEAL* includes all the homomorphic properties and key generation algorithms for BFV and CKKS cryptosystems defined in *Microsoft SEAL* library. The very first step when using *TenSEAL* is to define variable *context*, which contains the basic initialization parameters such as

- *Polynomial Modulus Degree* – denoted as $\mathcal{N}$ defining the degree of plaintext and ciphertext polynomials, as given in (2) and in (3). The value of $\mathcal{N}$ has to be a power of 2, for example $\mathcal{N} \in (1024, 2048, \dots, 32768)$. When choosing $\mathcal{N}$ we often face the speed vs. security compromise problem. The security level grows with increasing $\mathcal{N}$, but also the computational complexity of operations over ciphertexts. Thus the choice of $\mathcal{N}$ is not trivial and depends on the application use case.
- *Coefficient Moduli Chain* – is a vector of bit lengths of primes used to handle the size of ciphertext coefficients (from Eq. (20)). These primes are in cryptographic definitions often noted as q. The vector consisting of q sizes has the form of
$$\boldsymbol{\mathcal{Q}} = [\mathcal{Q}_\mathrm{S}, \underbrace{\mathcal{Q}_\mathrm{M}, \dots, \mathcal{Q}_\mathrm{M}}_{L}, \mathcal{Q}_\mathrm{S}] \tag{24}$$
where $\mathcal{Q}_\mathrm{S}$ are called *special* primes and middle primes $\mathcal{Q}_\mathrm{M}$ are used to handle the ciphertext coefficient size with L defining the number of $\mathcal{Q}_\mathrm{M}$. In *SEAL* implementation, each prime has to respect the restriction $\mathcal{Q}_\mathrm{M}, \mathcal{Q}_\mathrm{S} \leq 60\,\mathrm{bit}$. Finally, the sum of $\boldsymbol{\mathcal{Q}}$ elements is bounded by security level for defined $\mathcal{N}$ (Table 2).
- *Plaintext modulus* – noted as τ in BFV description. It represents the largest possible number or result to be sufficiently decrypted. The value of τ has to be chosen for the given $\mathcal{N}$ as
$$\mathrm{mod}\ (2\mathcal{N}, \tau) = 1. \tag{25}$$
with respect to condition $q > \tau$.
- *Scale* – the scaling factor Δ defines the space for messages in CKKS cryptographic scheme, as in (14). In *TenSEAL* the Δ is defined in bit length, i.e., $\Delta = (15, 20, 25, \dots)$.

Parameters $\mathcal{N}$ and τ are mandatory to choose for BFV scheme and $\mathcal{N}$ and Δ are mandatory parameters for CKKS configuration. Parameter $\boldsymbol{\mathcal{Q}}$ is optional and can be set for both schemes. If the vector $\boldsymbol{\mathcal{Q}}$ is not defined, *TenSEAL* library chooses default values of prime sizes to ensure the maximal possible level for given $\mathcal{N}$ with respect to security level λ. By keeping default values, the *TenSEAL* sets $\boldsymbol{\mathcal{Q}}$, such that a maximum number of homomorphic operations is allowed, which leads to unnecessary increased time for evaluation of homomorphic operations. From the perspective of control applications, where keeping all operations within one sample instant is of paramount importance, we aim to set the $\boldsymbol{\mathcal{Q}}$ as small as possible.

TABLE 2. Bit-security bounds on coefficient modulus bit-sizes in *SEAL*.

$\mathcal{N}$	λ	$\mathcal{Q}_\mathrm{max}$	$\mathcal{N}$	λ	$\mathcal{Q}_\mathrm{max}$
1024	128 bit	27 bits	8192	128 bit	218 bits
	192 bit	19 bits		192 bit	152 bits
	256 bit	14 bits		256 bit	118 bits
2048	128 bit	54 bits	16384	128 bit	438 bits
	192 bit	37 bits		192 bit	305 bits
	256 bit	29 bits		256 bit	237 bits
4096	128 bit	109 bits	32768	128 bit	881 bits
	192 bit	75 bits		192 bit	611 bits
	256 bit	58 bits		256 bit	476 bits

B. CKKS ENCODING & ENCRYPTION

This section presents and elaborates on inaccuracies caused by the encoding and encryption procedures in the CKKS approximate arithmetic. Specifically, we consider the following indicators:

- absolute error between the chosen testing vector $\mathbf{V}$ and its encoded-decoded result $\mathbf{V}_\mathrm{ecd}$ computed as $\sum |\mathbf{V} - \mathbf{V}_\mathrm{ecd}|$,
- absolute error between $\mathbf{V}$ and its encrypted-decrypted version $\mathbf{V}_\mathrm{ecd}$ defined by relation $\sum |\mathbf{V} - \mathbf{V}_\mathrm{ecd}|$.

where $\mathbf{V} = [2.5, 4, -5.6, 9.4, 18.9, -3.5]$. The testing procedure consisted of fixing one or two values of CKKS initialization parameters and increasing or decreasing the third one. The results are presented graphically at Figure 1 for encoding procedure and at Figure 2 for encryption.

The bar charts depicted in Figure 1 show the sums of absolute deviations between $\mathbf{V}$ and $\mathbf{V}_\mathrm{ecd}$ with changing setup parameters. The left chart at Figure 1 shows that with increasing polynomial modulus degree $\mathcal{N}$ the errors increase, but remain relatively small. In this case, we fixed the value of scale Δ and coefficient modulus sizes in $\mathcal{Q}$. Next, the middle graph of Figure 1 shows how the scale Δ affects the precision of encoding in CKKS for fixed values of $\mathcal{N}$ and $\boldsymbol{\mathcal{Q}}$. The higher the Δ, the more precise the encoding procedure is. By increasing the scale parameter, we also increase the space between signal and noise, thus the encoding procedure provides more precise results. Finally, when values of coefficient modulus primes in $\boldsymbol{\mathcal{Q}}$ are getting larger along with the scale Δ, the error between $\mathbf{V}$ and $\mathbf{V}_\mathrm{ecd}$ indistinctly decreases, which is confirmed by the right bar chart at Figure 1.

Next, we discuss errors introduced by encryption in CKKS. The results obtained from the testing procedure for encryption are graphically presented in Figure 2. Since encryption is the next step after encoding, there already will be a loss of accuracy from the encoding procedure. As depicted in Figure 2, the encryption procedure adds more inaccuracy to encrypted values. However, this amount of imprecision is negligible compared to the encoded messages. The high impact of scale on the precision of the original vector remains the same.

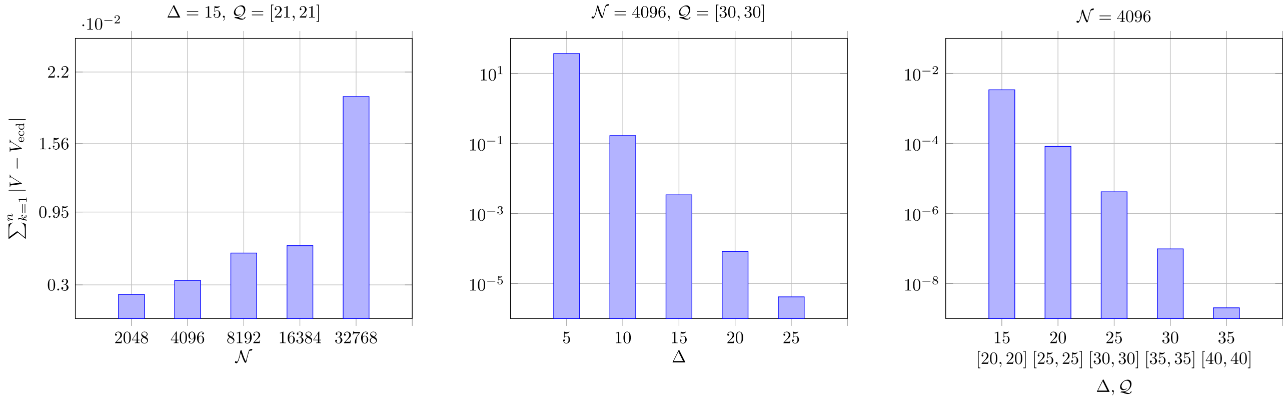

FIGURE 1. Bar charts of absolute deviations between vector V and its encoded-decoded version V_{ecd} resulting from testing procedure for encoding. Left: fixed Δ and 𝒬, middle: fixed 𝒩 and 𝒬, right: fixed 𝒩.

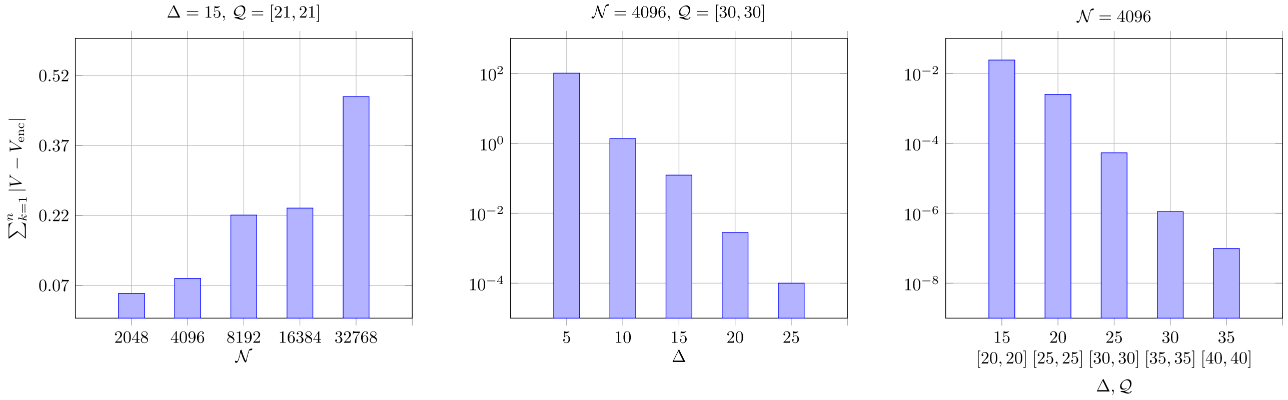

FIGURE 2. Bar charts of absolute deviations between vector V and its encrypted-decrypted form V_{enc} resulting from testing procedure for encryption. Left: fixed Δ and 𝒬, middle: fixed 𝒩 and 𝒬, right: fixed 𝒩.

C. BFV WITH FLOATING POINT NUMBERS

The BFV scheme operates with integers. Thus, to modify or transform the floating point number into an integer we can simply multiply the number by powers of 10 as

$$I = \lfloor F \cdot 10^{\theta} \rceil, \tag{26}$$

where I is integer, F is floating point number and θ is precision degree, usually equal to $1, 2, 3$ or higher. Note, that the final value of the I can be both positive and negative. There are limits on θ due to parameters $\mathcal{N}$ and τ, as shown later. One such parameter that bounds the value of θ is plaintext modulus τ, which defines the maximum value of an integer that can be encoded. The correct choice of θ is important mainly with respect to the plaintext modulus τ.

Consider an example of two floating point numbers $F_1 = 0.1587$ and $F_2 = 2.9879$. The aim is to homomorphically multiply F_1 by F_2. First, we transform F_1 and F_2 into integer form with (26) and the required precision degree θ as

$$I_1 = \lfloor F_1 \cdot 10^{\theta} \rceil, \tag{27a}$$
$$I_2 = \lfloor F_2 \cdot 10^{\theta} \rceil. \tag{27b}$$

TABLE 3. Table of plaintext modulus and precision degrees for BFV.

τ	θ	$\mathcal{D}(\mathcal{E}(I_1 I_2))$	$\lvert F_1 F_2 - \mathcal{D}(\mathcal{E}(I_1 I_2))\rvert$
65537	1	0.29	$1.842 \cdot 10^{-1}$
	2	0.447	$2.718 \cdot 10^{-2}$
1032193	3	0.4719	$2.234 \cdot 10^{-3}$
100073473	4	0.4742	$1.587 \cdot 10^{-5}$
10000039937	5	0.4742	$5.551 \cdot 10^{-17}$

Next, we encrypt I_1 and I_2 and homomorphically multiply them. Since the result from homomorphic multiplication, thus the precision degree θ is limited by plaintext modulus τ as

$$\tau > I_1 I_2, \tag{28}$$

we performed several experiments with several combinations of θ and τ to demonstrate their interdependence.

The results presented in Table 3 show that precision degree is dependent on plaintext modulus τ. From the final column, it is clear that with rising plaintext modulus τ we are able to get more precise results. However, the value of τ has to be chosen according to the BFV scheme polynomial modulus (25). In our case, we considered polynomial modulus $\mathcal{N} = 8192$ and used the default $\boldsymbol{\mathcal{Q}}$ for given $\mathcal{N}$ provided by *TenSEAL* library.

D. BFV & CKKS LEVELS

Before HE schemes will be included to control algorithms, we performed several settings and computational experiments to find the suitable scheme setup for the process control. Both BFV and CKKS are presented as leveled cryptographic schemes, where the number of levels is defined by a number of possible homomorphic multiplications over a single ciphertext until the relinearization procedure is able to recover the original message. This property is also referred to as multiplicative depth. The testing procedure for the acquisition of the multiplicative depth for each scheme was performed using the Algorithm 1, where $a = 1$, $a_{\mathrm{e}} = \mathcal{E}(1)$, $b_{\mathrm{e}} = \mathcal{E}(1)$ and the value of $\mathcal{M}$ represents the multiplicative depth. We define various combinations of parameters ($\mathcal{N}$,$\boldsymbol{\mathcal{Q}}$,Δ) for CKKS and ($\mathcal{N}$,$\boldsymbol{\mathcal{Q}}$,τ) for BFV and use the testing algorithm to define the maximal level of both schemes for different cryptographic settings. The numerical results are presented in Table 4.

Algorithm 1 Multiplicative Depth Test

```
$\mathcal{M} = 0$
while $a = \mathcal{D}(a_e)$ do
    $a_{\mathrm{e}} = a_{\mathrm{e}} b_{\mathrm{e}}$
    $\mathcal{M} = \mathcal{M} + 1$
end
```

Based on the results listed in Table 4 we conclude that the number of middle primes L defines the multiplicative depth $\mathcal{M}$ for given initialization parameters. Thus, we introduce the following steps leading to the correct setting of cryptographic parameters with respect to definitions in Section V-A, Table 4, and user requirements.

E. GUIDELINE FOR CONFIGURATION SETUP

For CKKS, we recommend the following procedure for selecting the moduli chain $\boldsymbol{\mathcal{Q}}$, scaling factor Δ, and corresponding polynomial modulus degree $\mathcal{N}$. Recall that sizes are expressed in a number of bits, and the guideline is as follows:

1) Select the desired binary precisions of messages' integer part $\eta_{\mathrm{I}} = \mathcal{Q}_{\mathrm{S}} - \mathcal{Q}_{\mathrm{M}}$ and decimal part $\eta_{\mathrm{D}} \approx 2\mathcal{Q}_{\mathrm{M}} - \mathcal{Q}_{\mathrm{S}}$.
2) Calculate the sizes of $\mathcal{Q}_{\mathrm{S}}$ and $\mathcal{Q}_{\mathrm{M}}$, such that $\mathcal{Q}_{\mathrm{M}} \geq 20$, $\mathcal{Q}_{\mathrm{S}} \geq \mathcal{Q}_{\mathrm{M}} + 10$, and $\mathcal{Q}_{\mathrm{M}}, \mathcal{Q}_{\mathrm{S}} \leq 60$.
3) Set scaling factor $\Delta = \mathcal{Q}_{\mathrm{M}}$.
4) Select a maximum multiplicative depth $\mathcal{M}$ of an arithmetic circuit to be the longest consecutive chain of homomorphic multiplications of a ciphertext propagating through the algorithm.
5) Calculate $\sum \boldsymbol{\mathcal{Q}} = 2\mathcal{Q}_{\mathrm{S}} + \mathcal{M}\mathcal{Q}_{\mathrm{M}}$.
6) For desired security level λ, find $\mathcal{N}$ in Table 2 such that $\sum \boldsymbol{\mathcal{Q}} \leq \mathcal{Q}_{\max}$.

TABLE 4. Table of BFV and CKKS multiplicative benchmarks.

Scheme	$\mathcal{N}$	τ	Δ	L	$\mathcal{Q}_{\mathrm{S}}$	$\mathcal{Q}_{\mathrm{M}}$	$\mathcal{M}$
BFV	4096	40961	-	1	30	20	1
	8192	65537	-	3	35	25	3
	16384	163841	-	5	40	30	5
	32768	786433	-	7	60	30	7
CKKS	4096	-	20	1	25	20	1
	8192	-	25	3	30	25	3
	16384	-	30	5	40	30	5
	32768	-	40	7	60	40	7

7) If $\mathcal{N}$ is too high, and therefore the setup is too computationally complex, the implementer can iterate through the procedure and select lower precision requirements.
8) On the other hand, if the computational burden is of little concern, and the difference between $\sum \boldsymbol{\mathcal{Q}}$ and $\mathcal{Q}_{\max}$ is considerable, the implementer can increase the precision of messages.

Example: Consider the required precision for integer part $\eta_{\mathrm{I}} = 14$ bits and decimal part $\eta_{\mathrm{I}} \approx 17$ bits. Solving the equations in the first point yields $\mathcal{Q}_{\mathrm{S}} = 45$ bits and $\mathcal{Q}_{\mathrm{M}} = 31$ bits. The scale Δ would be set to be the length of 31 bits. If the implementer needs to perform three consecutive multiplications, the multiplicative depth is $\mathcal{M} = 3$, which results in the sum of the moduli chain to be $\sum \boldsymbol{\mathcal{Q}} = 2 \times 45 + 3 \times 31 = 183$ bits. Table 2 shows that for 128-bit security level, the polynomial modulus degree is $\mathcal{N} = 8192$. Since the maximum sum of the moduli chain is 218 bits, the implementer still has 35 bits to allocate for setup. These can be used to increase the precision or extend multiplicative depth by another operation.

For selecting the moduli chain $\boldsymbol{\mathcal{Q}}$, plaintext modulus τ, and polynomial modulus $\mathcal{N}$ for BFV, we recommend the follow this procedure.

1) Select the desired precision degree θ to obtain integer form as in (26).
2) Define maximum multiplicative depth $\mathcal{M}$ given the application or user requirements.
3) For given $\mathcal{M}$, compute $\tau_0 = 10^{(\mathcal{M}+1)\theta}$
4) Calculate the sizes of $\mathcal{Q}_{\mathrm{S}}$ and $\mathcal{Q}_{\mathrm{M}}$, such that $\mathcal{Q}_{\mathrm{M}} \geq \log_2(\tau_0 10^3)$, $\mathcal{Q}_{\mathrm{S}} \geq \mathcal{Q}_{\mathrm{M}} + 10$, and $\mathcal{Q}_{\mathrm{M}}, \mathcal{Q}_{\mathrm{S}} \leq 60$.
5) Calculate $\sum \boldsymbol{\mathcal{Q}} = 2\mathcal{Q}_{\mathrm{S}} + \mathcal{M}\mathcal{Q}_{\mathrm{M}}$
6) For desired security level λ, find $\mathcal{N}$ in Table 2 such that $\sum \boldsymbol{\mathcal{Q}} \leq \mathcal{Q}_{\max}$.
7) Regarding $\mathcal{N}$, calculate plaintext modulus τ using the algorithm 2 initialized with τ_0.
8) If $\mathcal{N}$ is too high, and therefore the setup is too computationally complex, the implementer can iterate through the procedure and select lower precision requirements.
9) On the other hand, if the computational burden is of little concern, and the difference between $\sum \boldsymbol{\mathcal{Q}}$ and

Algorithm 2 Plaintext Modulus Generation

```
$\tau = \tau_0$
while  mod $(2\mathcal{N}, \tau) \neq 1$ do
    $\tau = \texttt{nextPrime}(\tau)$
end
```

$\mathcal{Q}_{\max}$ is considerable, the implementer can increase the precision of messages.

Note, that the function `nextPrime()` gives the nearest greater prime number to τ.

Example: Consider the required precision degree $\theta = 2$ and multiplicative depth $\mathcal{M} = 2$. Solving the equations in the third point yields $\tau_0 = 10^6$. According to limitations in point four, we get $\mathcal{Q}_{\text{M}} = 30$ and $\mathcal{Q}_{\text{S}} = 40$. If the implementer needs to perform two consecutive multiplications, the sum of the moduli chain is $\sum \boldsymbol{\mathcal{Q}} = 2\times40+2\times30 = 140$ bits. Table 2 shows that for 128-bit security level, the polynomial modulus degree is $\mathcal{N} = 8192$. For given $\mathcal{N}$, the algorithm 2 generates $\tau = 1032193$. Since the maximum sum of the moduli chain is 218 bits, the implementer still has 78 bits to allocate for setup. These can be used to increase the precision degree or extend multiplicative depth by another operation.

Recall that the bit length 60 comes from the *TenSEAL* library, as presented in Sec. V-A. Note that the definition of $\boldsymbol{\mathcal{Q}}$ is optional and not trivial. The user can omit the setting of $\boldsymbol{\mathcal{Q}}$ manually, and *TenSEAL* library will provide the default $\boldsymbol{\mathcal{Q}}$ ensuring the maximal possible level of the scheme for given $\mathcal{N}$ with regard to the security level. However, we recommend manually setting the $\boldsymbol{\mathcal{Q}}$ for direct control of the number of homomorphic multiplications and sizes of all primes.

VI. ENCRYPTED PROCESS CONTROL

This section presents the implementation of homomorphic crypto schemes in connection with a state-feedback control algorithm. We utilize the setup guideline from the previous section to define the cryptographic parameters for each cryptosystem regarding the encrypted control scenario. Here, the simulated measurement data are fully encrypted between the sensor of the process, the controller, and the actuator. Moreover, we also present a scenario where the state-feedback gain is also encrypted. From the technical point of view, the private key is stored only at the side of the process; hence no other elements of the closed-loop have access to the actual values of the simulated measurements.

A. LQR CONTROL

For this case study we worked with the model of the inverted pendulum. This process was chosen mainly because of its fast dynamics, thus to test the efficiency of the homomorphic crypto schemes with different setup parameters. The behavior of the inverted pendulum is described by discrete time state space model derived from nonlinear model [42] for the sampling period $T_{\text{s}} = 0.05$ s and defined as

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + Bu_k \tag{29}$$

where matrices A and B are

$$A = \begin{bmatrix} 1 & 0.0498 & 0.0034 & 0.0001 \\ 0 & 0.9909 & 0.1348 & 0.0034 \\ 0 & -0.0006 & 1.0392 & 0.0507 \\ 0 & -0.0229 & 1.5779 & 1.0392 \end{bmatrix} \tag{30a}$$

$$B = \begin{bmatrix} 0.0023 & 0.0908 & 0.0057 & 0.2292 \end{bmatrix}^{\top} \tag{30b}$$

and u_k is scalar control input at current step k. State vector $\mathbf{x}_k$ contains 4 states representing the physical quantities as follows:

x_1 – pendulum angle,
x_2 – pendulum angular velocity,
x_3 – cart position,
x_4 – cart linear velocity,

and u_k represents pendulum cart acceleration.

We present control strategy with the LQR controller using the homomorphic encryption properties to compute the state feedback control action respecting the control law

$$u_k = -\mathbf{K}\mathbf{x}_k \tag{31}$$

where the controller $\mathbf{K}$ was acquired by solving problem

$$\min \sum_{k=0}^{\infty} (\mathbf{x}_k^{\top} Q \mathbf{x}_k + u_k^{\top} R u_k), \tag{32a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = A\mathbf{x}_k + Bu_k \tag{32b}$$

for system matrices (30) and tuning factors

$$Q = \text{diag}\left(\begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}\right), \tag{33a}$$

$$R = 1. \tag{33b}$$

The feedback gain was of the following form

$$\mathbf{K} = \begin{bmatrix} -0.7277 & -1.2529 & 15.7967 & 2.9145 \end{bmatrix}^{\top} \tag{34}$$

Next step was to include BFV and CKKS cryptographic schemes in the state feedback control algorithm for two scenarios.

In the first scenario $\mathcal{S}_1$, the controller $\mathbf{K}$ was considered a public constant and only states $\mathbf{x}_k$ were encrypted. For the second scenario $\mathcal{S}_2$, both the states and the controller parameters were encrypted, thus we exploit the ciphertext-ciphertext multiplication property. We performed several experiments with various setups listed in Table 5 to demonstrate the impact of cryptographic parameters on the control quality and computational complexity for both scenarios. We also measured the storage sizes $\mathcal{V}(\cdot)$ of all keys. Configurations of HE schemes were defined according to guidelines described at the end of Section V-E.

The experiments were performed using algorithm 3, which represents the closed-loop encrypted state feedback control including presented HE schemes. Here, symbols p_k^x and p_k^u

TABLE 5. Table of various parameter setups and corresponding key sizes.

Scheme	Setup	$\mathcal{N}$	$\mathcal{Q}$	τ	θ	Δ	$\mathcal{V}(k_{\text{pub}})$	$\mathcal{V}(k_{\text{pvt}})$	$\mathcal{V}(k_{\text{gal}})$	$\mathcal{V}(k_{\text{eval}})$
BFV	$\mathcal{B}_1$	4096	$[30, 20, 30]$	40961	2	–	104 kB	51 kB	5 MB	208 kB
	$\mathcal{B}_2$	8192	$[40, 30, 40]$	10027009	3	–	270 kB	130 kB	13 MB	540 kB
	$\mathcal{B}_3$	16384	$[50, 40, 50]$	1000210433	4	–	654 kB	326 kB	34 MB	2 MB
CKKS	$\mathcal{C}_1$	4096	$[30, 16, 30]$	–	–	16	96 kB	47 kB	5 MB	192 kB
	$\mathcal{C}_2$	8192	$[40, 20, 40]$	–	–	20	251 kB	125 kB	12 MB	504 kB
	$\mathcal{C}_2$	16384	$[50, 30, 50]$	–	–	30	603 kB	300 kB	31 MB	2 MB

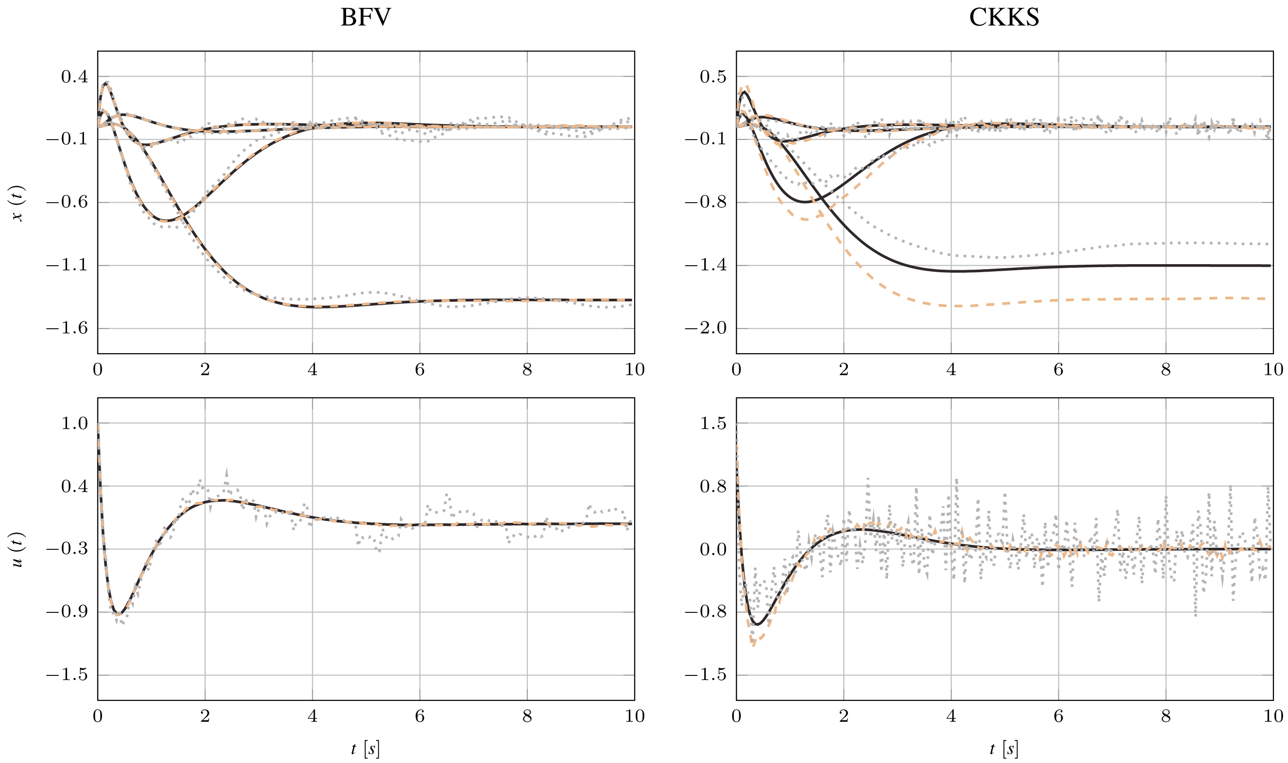

FIGURE 3. Control performances and inputs of encrypted control for the scenario $\mathcal{S}_1$ with public controller **K** and encrypted states $\mathbf{x}_k^{\mathcal{E}}$. Two upper graphs represent state behavior obtained by control inputs depicted at two lower graphs for each scheme. The subsequent lines represent encrypted control with the following setups from Table 5: Dotted line - $\mathcal{B}_1$ and $\mathcal{C}_1$, dashed line - $\mathcal{B}_2$ and $\mathcal{C}_2$, solid line - $\mathcal{B}_3$ and $\mathcal{C}_3$.

represent state and control input plaintexts, c_k^x and c_k^u denote state and control input ciphertexts and c^K marks controller ciphertext.

For each experiment, we computed the sum of control input errors $\mathcal{U} = \sum |u - u_{\text{enc}}|$, average ($\bar{t}$) and maximum ($\hat{t}$) evaluation times of control law over ciphertexts, and ciphertext binary size $\mathcal{V}(c)$. The graphical results are depicted at Figure 3 and Figure 4 and the corresponding numerical results are listed in Table 6.

The set of graphs in Figure 3 represent the results of encrypted control for scenario $\mathcal{S}_1$ with three setups for each scheme listed in Table 5. The associated numerical results are presented in Table 6. Setup $\mathcal{B}_1$ for BFV provides the highest sum of control input errors $\mathcal{U} = 16.19$ comparing to $\mathcal{B}_2$ and $\mathcal{B}_3$. This result is a consequence of low precision degree θ. On the other hand, setup $\mathcal{B}_1$ offers the fastest computations over ciphertexts ($\bar{t} = 1.5$ ms, $\hat{t} = 2.9$ ms) due to lowest modulus degree $\mathcal{N}$. The opposite to $\mathcal{B}_1$ is setup $\mathcal{B}_3$ providing the lowest $\mathcal{U} = 0.15$ but for the price of highest computational times ($\bar{t} = 6.8$ ms, $\hat{t} = 8.2$ ms). The more precise results are obtained thanks to higher θ, thus larger τ. However, the computations over ciphertexts are slower due to the higher value of $\mathcal{N}$. The setup $\mathcal{B}_2$ is somewhere between $\mathcal{B}_1$ and $\mathcal{B}_2$ in terms of computational times and also numerical results. For the CKKS, the results have similar characters when switching between setups $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. Here, the results are more precise for higher values of Δ, but the computations get slower with increasing $\mathcal{N}$. The ciphertext sizes $\mathcal{V}(c)$ are for the BFV twice as big as for CKKS, since BFV encodes and encrypts $\mathcal{N}$ slots, while CKKS only $\mathcal{N}/2$ slots.

The graphical results depicted in a group of charts at Figure 4 were obtained for scenario $\mathcal{S}_2$. The corresponding numerical results are listed in Table 6. For BFV setups $\mathcal{B}_1$,

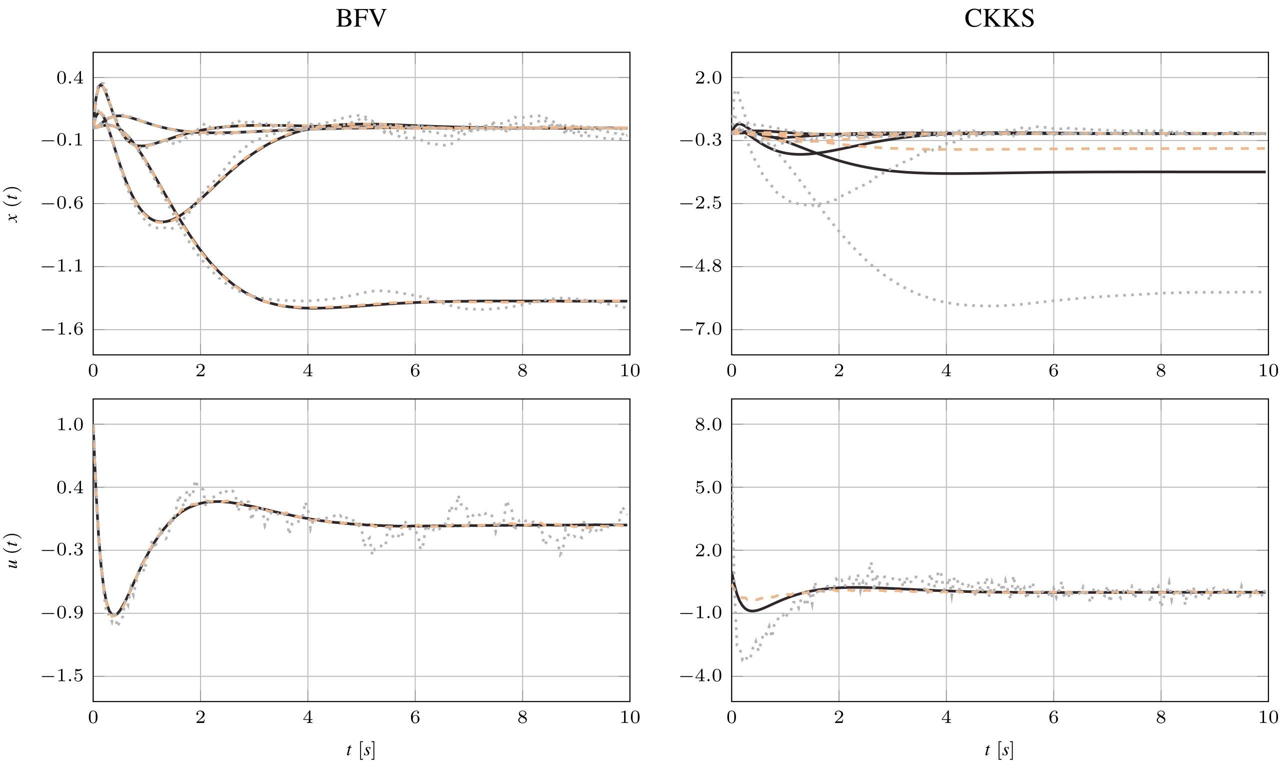

FIGURE 4. Control performances and inputs of encrypted control for the scenario $\mathcal{S}_2$ with encrypted controller $K^{\mathcal{E}}$ and encrypted states $\mathbf{x}_k^{\mathcal{E}}$. Two upper graphs represent state behavior obtained by control inputs depicted at two lower graphs for each scheme. The subsequent lines represent encrypted control with the following setups from Table 5: Dotted line - $\mathcal{B}_1$ and $\mathcal{C}_1$, dashed line - $\mathcal{B}_2$ and $\mathcal{C}_2$, solid line - $\mathcal{B}_3$ and $\mathcal{C}_3$.

Algorithm 3 Encrypted State Feedback Control

```
Process:
    Measure states:   $\mathbf{x}_k$
    Encode states:    $p_k^x = f_{\mathrm{E}}(\mathbf{x}_k)$
    Encrypt states:   $c_k^x = \mathcal{E}(p_k^x)$
Controller:
    if Scenario $\mathcal{S}_1$ then
        Evaluate control law:   $c_k^u = -\mathbf{K}c_k^x$
    else if Scenario $\mathcal{S}_2$ then
        Evaluate control law:   $c_k^u = c^K c_k^x$
    end
Process:
    Decrypt control input:   $p_k^u = \mathcal{D}(c_k^u)$
    Decode control input:    $u_k = f_{\mathrm{D}}(p_k^u)$
    Apply control input:     $u_k$
return
```

TABLE 6. Table of experimental results for both control scenarios.

Scenario	Scheme	Setup	$\mathcal{U}$	$\bar{t}$ [ms]	$\hat{t}$ [ms]	$\mathcal{V}(c)$
$\mathcal{S}_1$	BFV	$\mathcal{B}_1$	16.19	1.5	2.9	67 kB
		$\mathcal{B}_2$	1.56	3.1	6.1	171 kB
		$\mathcal{B}_3$	0.15	6.8	8.2	424 kB
	CKKS	$\mathcal{C}_1$	27.54	0.9	1.5	35 kB
		$\mathcal{C}_2$	7.68	1.9	3.0	92 kB
		$\mathcal{C}_3$	0.02	3.9	5.0	230 kB
$\mathcal{S}_2$	BFV	$\mathcal{B}_1$	16.19	3.8	5.0	66 kB
		$\mathcal{B}_2$	1.56	8.0	9.5	171 kB
		$\mathcal{B}_3$	0.15	17.1	18.3	422 kB
	CKKS	$\mathcal{C}_1$	99.68	1.3	2.5	35 kB
		$\mathcal{C}_2$	14.53	2.6	4.0	92 kB
		$\mathcal{C}_3$	0.03	5.5	6.5	230 kB

$\mathcal{B}_2$ and $\mathcal{B}_3$ the values of $\mathcal{U}$ remain the same due to similar precision degrees θ and plaintext modulus τ. What differs the scenario $\mathcal{S}_1$ from $\mathcal{S}_2$ are the computational times $\bar{t}$ and $\hat{t}$. Since for $\mathcal{S}_2$ we utilize the homomorphic multiplication over ciphertext, both $\bar{t}$ and $\hat{t}$ are more than two times higher due to the relinearization procedure that needs to be evaluated after each homomorphic multiplication. For the CKKS setups $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ we observe an increase of sum of control input errors $\mathcal{U}$. This growth is a consequence of errors emerging from encoding procedures based on approximate arithmetic. With increasing of scaling factor Δ the sum of control input errors $\mathcal{U}$ decreases since higher Δ provides more space for messages and the rounding error affects less the original message. Similar to BFV, the computational times $\bar{t}$ and $\hat{t}$ have grown due to the presence of relinearization procedure after homomorphic multiplication. However, both times remained almost two times smaller compared to BFV setups as in scenario $\mathcal{S}_1$. The ciphertext sizes $\mathcal{V}(c)$ remained at the same level as for scenario $\mathcal{S}_1$.

Overall, we conclude that from a computational complexity point of view, all of the presented setups were sufficient, thus their average and maximum computational times remained within the sampling period $T_s = 50\,\mathrm{ms}$. However, regarding the computed sum of control input errors and graphical results, we conclude that setups $\mathcal{B}_2$ and $\mathcal{B}_3$ for BFV or $\mathcal{C}_2$ and $\mathcal{C}_3$ for CKKS provide sufficient results in encrypted control of such a fast and unstable process like the inverted pendulum.

The presented results confirm that with increasing polynomial modulus degree $\mathcal{N}$ grows the computational complexity. However, higher $\mathcal{N}$ defines larger ring $\mathcal{R}$ providing more space for numerical precision (higher τ or Δ) and giving the possibility to perform several homomorphic multiplications (more middle primes in $\mathcal{Q}$).

VII. CONCLUSION

Homomorphic encryption is a promising technique for ensuring data security in the process control domain. We presented fully homomorphic encryption (FHE) methods that allow deploying controllers in any cloud services without revealing sensitive process data. We explored two principal FHE frameworks, BFV and CKKS, built above the RLWE approach. This paper provides a detailed guideline for implementing a state-feedback controller in the process control and the mechatronics domain. The overall success of closed-loop implementation with FHE depends on the proper selection of parameters. Therefore, we suggest following the guidelines in this paper for setting parameters to ensure that the encryption scheme provides the desired level of security, computational efficiency, and satisfactory control performance.

REFERENCES

[1] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and J. Dray, "Advanced encryption standard (AES)," Tech. Rep., 2001.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[3] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.

[4] J. Benaloh, "Dense probabilistic encryption," in *Proc. Workshop Sel. Areas Cryptogr.*, 1994, pp. 120–128.

[5] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT*, vol. 1592, J Stern, Ed., 1999, pp. 223–238.

[6] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, New York, NY, USA, May 2009, pp. 169–178.

[7] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," Cryptol. ePrint Arch., Tech. Rep. 2012/144, 2012.

[8] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 1–36, Jul. 2014.

[9] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachéne, "FHE: Fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, pp. 34–91, Jan. 2020.

[10] L. Ducas and D. Micciancio, "FHEW: Bootstrapping homomorphic encryption in less than a second," Cryptol. ePrint Arch., Tech. Rep. 2014/816, 2014.

[11] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*, T. Takagi and T. Peyrin, Eds. Cham, Switzerland: Springer, 2017, pp. 409–437.

[12] H. Karim and D. B. Rawat, "TollsOnly please—Homomorphic encryption for toll transponder privacy in Internet of Vehicles," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2627–2636, Feb. 2022.

[13] H. Yu, L. Yin, H. Zhang, D. Zhan, J. Qu, and G. Zhang, "Road distance computation using homomorphic encryption in road networks," *Comput., Mater. Continua*, vol. 69, no. 3, pp. 3445–3458, 2021.

[14] X. Yang, X. Yi, S. Nepal, A. Kelarev, and F. Han, "A secure verifiable ranked choice online voting system based on homomorphic encryption," *IEEE Access*, vol. 6, pp. 20506–20519, 2018.

[15] A. Wood, V. Shpilrain, K. Najarian, and D. Kahrobaei, "Private naive Bayes classification of personal biomedical data: Application in cancer data analysis," *Comput. Biol. Med.*, vol. 105, pp. 144–150, Feb. 2019.

[16] G. S. Çetin, H. Chen, K. Laine, K. Lauter, P. Rindal, and Y. Xia, "Private queries on encrypted genomic data," *BMC Med. Genomics*, vol. 10, no. 2, p. 45, Jul. 2017.

[17] H. Huang and L. Wang, "Efficient privacy-preserving face verification scheme," *J. Inf. Secur. Appl.*, vol. 63, Dec. 2021, Art. no. 103055.

[18] *IEEE Standard for Biometric Privacy*, Standard IEEE 2410–2021, Institute of Electrical and Electronics Engineers, Institute of Electrical and Electronics Engineers, 2021.

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

[20] L. Zhang, Y. Chen, and M. Li, "ADP-based remote secure control for networked control systems under unknown nonlinear attacks in sensors and actuators," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6003–6014, Sep. 2022.

[21] S. Zhou, Z. Yu, E. S. A. Nasr, H. A. Mahmoud, E. M. Awwad, and N. Wu, "Homomorphic encryption of supervisory control systems using automata," *IEEE Access*, vol. 8, pp. 147185–147198, 2020.

[22] M. Miyamoto, K. Teranishi, K. Emura, and K. Kogiso, "Cybersecurity-enhanced encrypted control system using keyed-homomorphic public key encryption," *IEEE Access*, vol. 11, pp. 45749–45760, 2023.

[23] J. Tran, F. Farokhi, M. Cantoni, and I. Shames, "Implementing homomorphic encryption based secure feedback control," *Control Eng. Pract.*, vol. 97, Apr. 2020, Art. no. 104350.

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

[25] M. Ruan, H. Gao, and Y. Wang, "Secure and privacy-preserving consensus," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 4035–4049, Oct. 2019.

[26] M. S. Darup, A. Redder, and D. E. Quevedo, "Encrypted cooperative control based on structured feedback," *IEEE Control Syst. Lett.*, vol. 3, no. 1, pp. 37–42, Jan. 2019.

[27] A. B. Alexandru, K. Gatsis, Y. Shoukry, S. A. Seshia, P. Tabuada, and G. J. Pappas, "Cloud-based quadratic optimization with partially homomorphic encryption," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2357–2364, May 2021.

[28] A. B. Alexandru, M. Morari, and G. J. Pappas, "Cloud-based MPC with encrypted data," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 5014–5019.

[29] M. S. Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted MPC for linear constrained systems," *IEEE Control Syst. Lett.*, vol. 2, no. 2, pp. 195–200, Apr. 2018.

[30] C. Gouert, R. Khan, and N. G. Tsoutsos, "Optimizing homomorphic encryption parameters for arbitrary applications," *Cryptol. ePrint Arch.*, to be published.

[31] J. Cabrero-Holgueras and S. Pastrana, "Towards automated homomorphic encryption parameter selection with fuzzy logic and linear programming," *Expert Syst. Appl.*, vol. 229, Nov. 2023, Art. no. 120460.

[32] V. Herbert, "Automatize parameter tuning in ring-learning-with-errors-based leveled homomorphic cryptosystem implementations," Cryptol. ePrint Arch., Tech. Rep., 2019/1402, 2019. [Online]. Available: https://eprint.iacr.org/2019/1402

[33] M. Yagisawa, "Fully homomorphic encryption without bootstrapping," *IACR Cryptol. EPrint Arch.*, vol. 2015, p. 474, Jan. 2015.

[34] (May 2013). *Helib v2.2.1*. [Online]. Available: https://github.com/homenc/HElib

[35] (Aug. 2022). *Lattigo v4*, EPFL-LDS, Tune Insight SA. [Online]. Available: https://github.com/tuneinsight/lattigo

[36] *Microsoft SEAL (Release 3.6)*, Microsoft Research, Redmond, WA, USA, Nov. 2020. [Online]. Available: https://github.com/Microsoft/SEAL

[37] S. M. Anggriane, S. M. Nasution, and F. Azmi, "Advanced e-voting system using Paillier homomorphic encryption algorithm," in *Proc. Int. Conf. Informat. Comput. (ICIC)*, Oct. 2016, pp. 338–342.

[38] W. Ding, Z. Yan, and R. H. Deng, "Encrypted data processing with homomorphic re-encryption," *Inf. Sci.*, vols. 409–410, pp. 35–55, Oct. 2017.

[39] M. Furka, K. Kiš, M. Klauco, and M. Kvasnica, "Usage of homomorphic encryption algorithms in process control," in *Proc. 23rd Int. Conf. Process Control (PC)*, Jun. 2021, pp. 43–48.

[40] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, "Homomorphic encryption security standard," HomomorphicEncryption.org, Toronto, ON, Canada, Tech. Rep., Nov. 2018.

[41] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "TenSEAL: A library for encrypted tensor operations using homomorphic encryption," Tech. Rep., 2021.

[42] P. Bakarác, M. Kalúz, and L. Cirka, "Design and development of a low-cost inverted pendulum for control education," in *Proc. 21st Int. Conf. Process Control (PC)*, Jun. 2017, pp. 398–403.

M. FURKA received the master's degree in process control from the Slovak University of Technology in Bratislava, where he is currently pursuing the Ph.D. degree with the Department of Information Engineering and Process Control. His research interest includes the implementation of homomorphic encryption frameworks in secured process control strategies.

M. KALÚZ received the M.Sc. and Ph.D. degrees in process control from the Slovak University of Technology in Bratislava (STUBA), in 2010 and 2014, respectively. Currently, he is a Postdoctoral Researcher and an Assistant Professor with STUBA. His research interests include control education, information technologies, and process control security.

M. FIKAR received the M.E. and Ph.D. degrees in chemical engineering from the Slovak University of Technology in Bratislava, in 1989 and 1994, respectively. He has stayed with the Faculty of Chemical and Food Technology, STUBA, where he is currently a Professor and the Institute Director. He was a Postdoctoral Fellow in Nancy, France, Alexander von Humboldt Fellow in Bochum, Germany, and has spent several stays in Denmark, Germany, France, and Switzerland. He is the coauthor of two international monographs, 80 journal articles, and more than 200 peer-reviewed conference publications. His current research interests include optimal control, MPC, and chemical process control.

M. KLAUČO (Member, IEEE) received the M.Sc. degree in automation from Denmark Technical University and the master's degree in process control from the Slovak University of Technology in Bratislava (STUBA), where he is currently pursuing the Ph.D. degree (summa cum laude) in process control. He is also an Associate Professor in cybernetics and the Head of the Department of Information Engineering and Process Control, STUBA. His current research interests include applications of optimal control and machine learning in connection with secure control.