A Verifiable Computing Scheme for Encrypted Control Systems

F. Stabile, W. Lucia, *Member, IEEE*, A. Youssef, *Member, IEEE* and G. Franzè , *Senior Member, IEEE*

***Abstract*— The proliferation of cloud computing technologies has paved the way for deploying networked encrypted control systems, offering high performance, remote accessibility and privacy. However, in scenarios where the control algorithms run on third-party cloud service providers, the control's logic might be changed by a malicious agent on the cloud. Consequently, it is imperative to verify the correctness of the control signals received from the cloud. Traditional verification methods, like zero-knowledge proof techniques, are computationally demanding in both proof generation and verification, may require several rounds of interactions between the prover and verifier and, consequently, are inapplicable in real-time control system applications. In this paper, we present a novel computationally inexpensive verifiable computing solution inspired by the probabilistic cut-and-choose approach. The proposed scheme allows the plant's actuator to validate the computations accomplished by the encrypted cloud-based networked controller without compromising the control scheme's performance. We showcase the effectiveness and real-time applicability of the proposed verifiable computation scheme using a remotely controlled Khepera-IV differential-drive robot.**

***Index Terms*— Verifiable computing, cloud-based control, encrypted control**

I. Introduction

CCLOUD-based control systems [1] exploit the well-established advantages of cloud and edge computing, such as increased reliability, easier scalability and reduced IT expenses to enhance the performance of various Cyber-Physical Systems (CPSs). However, these cloud-based systems hinge on communication links for exchanging measurement data and control signals between the plant and the cloud-based controller. Thus, the early research in the field of CPS security has been mainly centered on safeguarding these systems from network attackers with access to the measurement and/or control channels [2]. Various control theory based approaches were proposed to mitigate the vulnerability to false data injection attacks [3]. To ensure the privacy of the computation at the cloud controller, homomorphic encryption (HE) schemes were proposed to implement polynomial control laws on encrypted data [4], [5], [6]. On the other hand, given the safety-critical nature of such systems, it is also necessary to provide the physical plant/smart actuator with an efficient mechanism to validate the correctness of the computations conducted by the cloud-based controller on encrypted data. Verifiable computing aims not only to obtain the control input computed by the cloud and a proof of its correctness, but also to enable the plant/smart actuator to verify this proof with significantly less computational effort than calculating the control input from scratch. The simplest, yet least efficient, approach for verifiable computation is replication. In this scenario, the verifier (pant/actuator) utilizes multiple, presumably non-colluding cloud-based controllers to perform identical computation tasks. Upon receiving a minimum consensus of results from these multiple controllers, the verifier assumes those results are correct. An alternative method to ensure the integrity of the computation at the cloud based controller employs a hardware-based Trusted Execution Environment (TEE) such as Intel SGX, as demonstrated in [7]. Nevertheless, the latest attacks on SGX have revealed that hardware remains vulnerable to compromise and this could lead to jeopardize both the TEE and the overall systems' security. Although advanced cryptographic techniques for verifiable computation(e.g., see [8]–[10], can be used to verify the integrity of outsourced computations, integrating them with the complex structure of HE schemes remains a challenge [11], [12]. The (Boolean) circuit representations for control theory algorithms are too complex to implement using these techniques. Additionally, certain techniques like interactive proofs demand multiple rounds of communication between prover and verifier. In the context of CPS, Cheon et al. [13] proposed a verifiable computation scheme which enables the actuator to verify the non-encrypted controller's computation. However their scheme is only applicable to linear systems and it cannot be used for encrypted control scenarios. Mahfouzi et al. [14] presented a proof of concept of an industrial controller in a cloud-based verifiable computation framework using the Pequin, an end-to-end toolchain for verifiable computation, SNARKs, and probabilistic proofs. However, the authors did not consider preserving the confidentiality of messages transmitted between the local controller and the cloud.

A. Paper's contribution

The state-of-the-art lacks solutions capable of detecting attacks against the integrity of cloud-based encrypted con-

The work was supported by Mitacs under the grant IT34199.

Francesca Stabile and Giuseppe Franzè are with DIMEG, Università della Calabria, Via Pietro Bucci, Cubo 42-C, Rende (CS), 87036, Italy, stabilefrancesca99@outlook.com, giuseppe.franze@unical.it.

Walter Lucia and Amr Youssef are with the Concordia Institute for Information Systems Engineering (CIISE), Concordia University, Montreal, Canada, walter.lucia@concordia.ca, youssef@ciise.concordia.ca.

trollers. In this paper, we present a verifiable computing solution inspired by the cut-and-choose cryptographic technique, commonly employed in secure multiparty computation protocols [15], and cut-and-choose auditing [16]. In this approach, the party wishing to perform a secure computation first generates a set of computations and then sends them to a verifier, who proceeds to randomly select a subset from the set and asks the party to disclose the input and output of each computation. In the event that the computations within the chosen subset prove to be correct, the verifier can trust that the remaining computations are correct too, enabling the party to proceed with the secure computation. However, if any of the computations in the subset are found to be incorrect, the protocol is terminated. Our contributions can be summarized as follows.

- We propose a practical verifiable computing solution for encrypted control systems. The proposed probabilistic scheme allows the plant's actuator to efficiently validate the computations accomplished by the cloud-based encrypted networked controller without compromising the control system's performance. Moreover, we prove that the proposed solution ensures asymptotic detection of any integrity attack. These results are achieved by leveraging two essential ingredients: (i) an offline pre-computed set of control signals complying with the control law for a given set of decoy input parameters, and (ii) a semantically secure encryption mechanism which does not allow the adversary to distinguish the decoys from the actual measurements/control inputs.
- We experimentally validate the performance and effectiveness of the proposed approach using a remotely maneuvered Khepera IV robot[1].

II. Preliminaries and Problem Formulation

Given a vector v and a matrix M, v_i denotes the $i-th$ element of v and $M_{i,j}$ the $(i,j)-$entry of M. Moreover, I_2 denotes a 2×2 identity matrix while $\mathbf{1}_2$ and $\mathbf{0}_2$ denote column vectors of size two where both elements are equal to 1 and 0, respectively. Given a number z in the message space of a cryptosystem, we denote the encryption and decryption procedures as $\mathtt{E}[z]$ and $\mathtt{D}[z]$, respectively. Consider a random variable $\mathcal{X}$ with binary possible outcomes "*success*" and "*fail*." The probability of getting a "*success*" in a single trial is denoted as p, while the probability of $k+1$ successes in $k+1$ binary independent experiments is denoted as $p_{[0,\,k]}$.

Definition 1: Homomorphic Encryption (HE) refers to a particular class of encryption mechanisms that enables mathematical operations (limited in their type and/or number) to be carried out directly on encrypted data. □

Definition 2: Consider two arbitrary numbers, z_1 and z_2, in the message space of the cryptosystem. A cryptosystem is said to be *multiplicatively homomorphic* if there exists an operation "$\otimes$" allowing encrypted multiplications, i.e., $z_1 z_2 = \mathtt{D}[\mathtt{E}[z_1]\otimes\mathtt{E}[z_2]]$. Analogously, a cryptosystem is said to be *additively homomorphic* if there exists an operation "$\oplus$" such that encrypted additions can be performed, i.e., $z_1+z_2 = \mathtt{D}[\mathtt{E}[z_1]\oplus\mathtt{E}[z_2]]$.

[1] http://www.k-team.com/khepera-iv

A. Networked control system setup

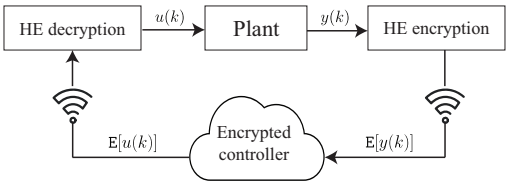


Fig. 1: Cloud-based encrypted control scheme

Of interest is the class of networked control systems shown in Fig. 1, where the plant's dynamic is described by the discrete-time model

$$x(k+1) = f(x(k),u(k)), \quad y(k) = g(x(k),u(k)) \tag{1}$$

with $k\in\mathbb{Z}_+ = \{0,1,\ldots\}$ the discrete-time index, $x\in\mathbb{R}^n, u\in\mathbb{R}^m, y\in\mathbb{R}^p$ the state, input, and output vectors, and $f:\mathbb{R}^n\times\mathbb{R}^m\to\mathbb{R}^n$, $g:\mathbb{R}^n\times\mathbb{R}^m\to\mathbb{R}^p$. Moreover, a semantically secure [17] randomized HE scheme is used to transmit signals between the plant and the cloud-based controller, allowing the control logic to be executed directly on the encrypted data. The considered dynamic output feedback tracking controller presents the following plaintext dynamic

$$\begin{array}{rcl} x_c(k+1) & = & f_c(x_c(k),y(k),r(k)) \\ u(k) & = & g_c(x_c(k),y(k),r(k)) \end{array} \tag{2}$$

where $x_c\in\mathbb{R}^{n_c}$ is the state of the controller, $r(k)\in\mathbb{R}^p$ the reference signal, and $f_c:\mathbb{R}^{n_c}\times\mathbb{R}^p\times\mathbb{R}^p\to\mathbb{R}^{n_c}$, $g_c:\mathbb{R}^{n_c}\times\mathbb{R}^p\times\mathbb{R}^p\to\mathbb{R}^m$ the control strategy.

Assumption 1: The control (2) can be implemented on the encrypted data. □

In what follows, the encrypted-version of (2) is generically described by:

$$\begin{array}{rcl} \mathtt{E}[x_c(k+1)] & = & f_c^E(\mathtt{E}[x_c(k)],\mathtt{E}[y(k)],\mathtt{E}[r(k)]) \\ \mathtt{E}[u(k)] & = & g_c^E(\mathtt{E}[x_c(k)],\mathtt{E}[y(k)],\mathtt{E}[r(k)]) \end{array} \tag{3}$$

where $f_c^E(\cdot,\cdot,\cdot)$ and $g_c^E(\cdot,\cdot,\cdot)$ denote the encrypted controller's operations corresponding to $f_c(\cdot,\cdot,\cdot)$ and $g_c(\cdot,\cdot,\cdot)$.

B. Threat model and Problem formulation

An adversary wants to replace the executed encrypted control operations (3), i.e.,

$$\left\{f_c^E(\cdot,\cdot,\cdot), g_c^E(\cdot,\cdot,\cdot)\right\} \to \left\{f_a^E(\cdot,\cdot,\cdot), g_a^E(\cdot,\cdot,\cdot)\right\} \tag{4}$$

where $\left\{f_a^E(\cdot,\cdot,\cdot), g_a^E(\cdot,\cdot,\cdot)\right\} \neq \left\{f_c^E(\cdot,\cdot,\cdot), g_c^E(\cdot,\cdot,\cdot)\right\}$ define the adversary encrypted control logic. Consequently, the encrypted $\mathtt{E}[u(k)]$ and $\mathtt{E}[x_c(k+1)]$ signals received by the plant from the cloud might be the result of the adversary control logic (4). In this work, we assume that the attacker's actions do not force the plant to reach a configuration outside of working domain of the legitimate control law (3). Consequently, the considered class of attacks is sufficiently intelligent to be

stealthy against safety-preserving mechanisms on the plant's side, see e.g., [18].

***Problem** 1: Given the encrypted networked control architecture shown in Fig. 1, design a secure verifiable computing solution capable of assessing the integrity of the encrypted control logic (3). Moreover, the solution must perform computations in real-time, avoid interference with control actions, and probabilistically ensure the absence of stealthy control logic alterations over time.*

III. Proposed solution

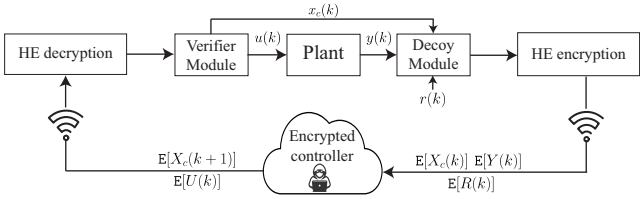


Fig. 2: Proposed verifiable computing scheme for encrypted control systems.

The proposed solution (see Fig. 2) is inspired by the cut-and-choose cryptographic technique, commonly employed in secure multiparty computation protocols [15]. For the considered encrypted networked control system setup, we customize the cut-and-choose paradigm as follows:

- The cloud-based encrypted controller is the party performing the agreed upon encrypted control system logic, as described by equation (3).
- A decoy module, local to the plant, is in charge of generating a set of $n_d > 0$ decoy (e.g., artificial) measurement vectors, namely $y_d^i(k)$, $i = 1, \dots, n_d$. The n_d+1 measurement vectors (decoys + actual measurement) are encrypted and sent to the controller. The controller executes (3) for each received measurement vector and produces a set of $n_d + 1$ computations (i.e., encrypted control inputs).
- The verifier is a subsystem local to the plant checking the validity of the decrypted control input vectors obtained from the cloud for all n_d decoy measurement vectors (disclosed to the verifier by the decoy subsystem). If all the decoy computations are deemed accurate, then $u(k)$ is assumed correct and applied to the plant.

Moreover, to ensure that the proposed decoy-based verifiable computing paradigm provides a solution to Problem 1, the following objectives must be achieved:

(O_1) The verification of the computations for each decoy must be real-time affordable and not require the online local re-computation of the control algorithm itself.

(O_2) The decoy measurements should not interfere with the computation of the control action $\mathbb{E}[u(k)]$ for the actual sensor measurements $\mathbb{E}[y(k)]$.

(O_3) The attacker's probability $p_{[0,\,k]}$ of changing the control logic as in (4) without being detected asymptotically converges to zero, i.e. $\lim_{k\to\infty} p_{[0,\,k]} = 0$.

A. Proposed verifiable computing scheme

In what follows, first the proposed verifiable computing scheme is presented and then its effectiveness in addressing the points (O_1)-(O_3) is proven.

∘ For each required computation, the encrypted controller receives from the plant the set $\{\mathbb{E}[x_c(k)], \mathbb{E}[y(k)], \mathbb{E}[r(k)]\}$. The encrypted controller first resets its state to $\mathbb{E}[x_c(k)]$, then computes the updated internal state $\mathbb{E}[x_c(k+1)]$ and control input $\mathbb{E}[u(k)]$ which are both transmitted to the plant. If the computation is pertaining to the real measurement, the plaintext corresponding to $\mathbb{E}[x_c(k)]$ is equal to the one provided by the controller at the previous iteration.

∘ Offline, a set $\mathcal{D}$ of N_d tuples $(\bar{u}_d, \bar{x}_{c_d}^+, \bar{x}_{c_d}, \bar{y}_d, \bar{r}_d)$, complying with the controller's logic (2) is defined, i.e.,

$$\mathcal{D} = \{(\bar{u}_d^j, \bar{x}_{c_d}^{j^+}, \underbrace{\{\bar{x}_{c_d}^j, \bar{y}_d^j, \bar{r}_d^j\}}_{\text{decoy}}), j = 1, \dots, N_d : \underbrace{\bar{u}_d^j = g_c(\bar{x}_{c_d}^j, \bar{y}_d^j, \bar{r}_d^j)}_{\text{expected control input}}\}, \underbrace{\bar{x}_{c_d}^{j^+} = f_c(\bar{x}_{c_d}^j, \bar{y}_d^j, \bar{r}_d^j)}_{\text{expected next state}}\} \tag{5}$$

∘ At each $k \geq 0$, a set of $n_d \geq 1$ decoys is selected (with possible repetitions) from $\mathcal{D}$ and freshly encrypted into $\{\mathbb{E}[x_{c_d}^i(k)], \mathbb{E}[y_d^i(k)], \mathbb{E}[r_d^i(k)]\}_{i=1}^{n_d}$.

∘ At each $k \geq 0$ the n_d decoys are randomly arranged with the actual vector $\{x_c(k), y(k), r(k)\}$ to form the following encrypted matrices which are sent to the controller (see Fig. 2):

$$\begin{aligned} \mathbb{E}[X_c(k)] &:= \left[\mathbb{E}[x_c(k)], \mathbb{E}[x_{c_d}^1(k)], \dots, \mathbb{E}[x_{c_d}^{n_d}(k)]\right] \Omega_k \\ \mathbb{E}[Y(k)] &:= \left[\mathbb{E}[y(k)], \mathbb{E}[y_d^1(k)], \dots, \mathbb{E}[y_d^{n_d}(k)]\right] \Omega_k \\ \mathbb{E}[R(k)] &:= \left[\mathbb{E}[r(k)], \mathbb{E}[r_d^1(k)], \dots, \mathbb{E}[r_d^{n_d}(k)]\right] \Omega_k \end{aligned} \tag{6}$$

with $\Omega(k) \in \mathbb{R}^{(n_d+1)\times(n_d+1)}$ a random permutation matrix. On the other hand, the controller, without the knowledge of Ω_k, performs the encrypted control algorithm (3) for each column of the received matrices and arranges the outputs into the following, which are sent to the plant (see Fig. 2):

$$\begin{aligned} \mathbb{E}[U(k)] &:= [\mathbb{E}[u(k)], \mathbb{E}[u_d^1(k)], \dots, \mathbb{E}[u_d^{n_d}(k)]]\Omega_k \\ \mathbb{E}[X_c(k+1)] &:= [\mathbb{E}[x_c(k+1)], \mathbb{E}[x_{c_d}^1(k+1)], \\ &\qquad \dots, \mathbb{E}[x_{c_d}^{n_d}(k+1)]]\Omega_k \end{aligned} \tag{7}$$

∘ The verifier decrypts (7) and checks, for each used decoy $\{x_{c_d}^i(k), y_d^i(k), r_d^i(k)\}_{i=1}^{n_d}$, if the computed control inputs and next states are equal to the expected values $\{u_d^i(k)\}_{i=1}^{n_d}$ and $\{x_{c_d}^i(k+1)\}_{i=1}^{n_d}$ in (5).

***Proposition** 1: If $N_d \geq 2$, the proposed verifiable computing scheme achieves the objectives (O_1)-(O_3).*

Proof: The proof can be divided into three parts:

- (O_1) : by constructions, the decoy set (5) is pre-computed. Therefore, for each used decoy $\{\mathbb{E}[x_{c_d}^i(k)], \mathbb{E}[y_d^i(k)], \mathbb{E}[r_d^i(k)]\}$, the verification of $u_d^i(k)$ and $x_{c_d}^i(k+1)$ prescribes only their comparison with the pre-computed value in (5). Consequently, the computational overhead of the proposed solution is mainly related to the encryption and decryption of the decoys.

- (O_2) : The controller resets the internal state before each computation. Moreover, for the actual measurement, the reset value is equal to the controller's state at the previous iteration. Consequently, regardless of n_d, N_d, Ω_k, the computations pertaining to the decoys do not affect $\mathbb{E}[u(k)]$.

- (O_3) : If the attacker is only interested in arbitrarily affecting the control logic and $n_d = 1, N_d = 1$ (i.e., 1 decoy from a pool of a single decoy), then the attacker can guess at $k = 0$ the decoy measurement vector and use the valid decoy output $\mathtt{E}[u_d^1(0)]$ to send to the plant the matrix $\mathtt{E}[U(k)] = [\mathtt{E}[u_d^1(0)], \mathtt{E}[u_d^1(0)]]\Omega_k$, $\forall\, k$ which would trigger an anomaly with probability 0.5, irrespective of k. To avoid copying the same ciphertext, which could lead to trivial detection, the attacker can re-encrypt it. On the other hand, for $N_d = 2$, or any other greater value, the probability that an attacker can, in a single trial, successfully change $\mathtt{E}[u(k)]$ while leaving the decoy computation unchanged is equal to the probability of guessing in which column of $\mathtt{E}[Y(k)]$ the measurement vector $\mathtt{E}[y(k)]$ is contained. Given the randomized nature of the used homomorphic encrypting scheme and Ω_k, each column vector has an equal probability, resulting in a Bernoulli probability of cheating $p = 1/(n_d + 1)$. Consequently, in the interval $[0, k]$, the probability $p_{[0,\,k]}$ that the attacker remains always undetected follows the Binomial distribution $(\frac{1}{n_d+1})^{k+1}$ that asymptotically converges to zero as $k \to \infty$. ■

IV. Proof-of-concept validation using a remotely controlled mobile robot

In what follows, by considering as case of study a remotely controlled differential-drive robot, we verify the effectiveness of the proposed verifiable computing scheme.

A. Robot Model

We consider a differential-drive robot equipped with two real independently driven wheels and a front castor wheel for body support. The pose of the robot is described by the planar coordinates (p_x, p_y) of its center of mass and orientation θ. By resorting to the forward Euler discretization method and a sampling time $t_s > 0$, the discrete-time kinematic model of the differential-drive is:

$$\begin{aligned} p_x(k+1) &= p_x(k) + \tfrac{t_s R}{2}\cos\theta(k)(\omega_r(k) + \omega_l(k)) \\ p_y(k+1) &= p_y(k) + \tfrac{t_s R}{2}\sin\theta(k)(\omega_r(k) + \omega_l(k)) \\ \theta(k+1) &= \theta(k) + \tfrac{t_s R}{D}(\omega_r(k) - \omega_l(k)) \end{aligned} \tag{8}$$

where $R > 0$ is the radius of the wheels, $D > 0$ the rear axle length, and $u^D = [\omega_r,\, \omega_l]^T \in \mathbb{R}^2$ the control input vector, which consists of the angular velocities of the right and left wheel, respectively. Moreover, $x(k) = [p_x(k), p_y(k), \theta(k)]^T \in \mathbb{R}^3$ denotes the robot's state vector.

By denoting with $v(k)$ and $\omega(k)$ the linear and angular velocities of the center of mass of the robot, it is possible to apply to (8) the static transformation

$$\begin{bmatrix} v(k) \\ \omega(k) \end{bmatrix} = H \begin{bmatrix} \omega_r(k) \\ \omega_l(k) \end{bmatrix}, \quad H := \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{D} & \frac{-R}{D} \end{bmatrix}, \tag{9}$$

and describe the robot's behavior by means of the following discrete-time unicycle model:

$$\begin{aligned} p_x(k+1) &= p_x(k) + t_s v(k)\cos\theta(k) \\ p_y(k+1) &= p_y(k) + t_s v(k)\sin\theta(k) \\ \theta(k+1) &= \theta(k) + T\omega(k) \end{aligned} \tag{10}$$

Since (10) has been obtained using an Euler forward discretization and a static transformation, we can exploit the commutative property between feedback linearization and the input-output linearization used in [19, Property 1] to linearize the discrete-time model of the unicycle. In particular, by considering a small scalar $b > 0$ and two virtual outputs

$$y(k) = \left[\; p_x(k) + b\cos\theta(k),\, p_y(k) + b\sin\theta(k) \;\right]^T, \tag{11}$$

representing the coordinates of a fictitious point $\mathbf{B}$ displaced at a distance b from $[p_x, p_y]^T$, the state-feedback law

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = T_{FL}(\theta)\begin{bmatrix} u_1 \\ u_2 \end{bmatrix},\; T_{FL}(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ \frac{-\sin\theta}{b} & \frac{\cos\theta}{b} \end{bmatrix} \tag{12}$$

recasts (10) into a two-single integrator model with a decoupled nonlinear internal dynamics [19, Property 1]:

$$y(k+1) = Ay(k) + Bu(k),\; A = I_2,\; B = TI_2 \tag{13a}$$

$$\theta(k+1) = \theta(k) + T\frac{-\sin\theta(k)u_1(k) + \cos\theta(k)u_2(k)}{b} \tag{13b}$$

where $u(k) = [u_1(k),\, u_2(k)]^T \in \mathbb{R}^2$ are the control inputs of the feedback-linearized robot model. As stated in [19, Remark 1], any linear tracking controller for (11) allows $y(k)$ to track any reference trajectory with a stable internal dynamics for $\theta(k)$.

B. HE cryptosystem

The data exchanged with the controller are encrypted using the Paillier cryptosystem [20]. The Paillier cryptosystem is a randomized and semantically secure partially homomorphic encryption scheme which allows encrypted additions of two ciphertexts (see Definition 2) and multiplications of a ciphertext by a plaintext number, i.e., $z_1 z_2 = \mathtt{D}[\mathtt{E}[z_1] \otimes z_2]$.

C. Tracking controller

Consider a desired robot's reference pose $p^r(k) = [p_x^r(k), p_y^r(k), \theta^r(k)]^T \in \mathbb{R}^3, \forall\, k \geq 0$ and the associated 2D reference trajectory, namely $r^B(k)$, for the $\mathbf{B}$ point

$$r(k) = [p_x^r(k) + b\cos\theta^r(k),\, p_y^r(k) + b\sin\theta^r(k)]^T,\; \forall\, k \geq 0. \tag{14}$$

A standard discrete-time Proportional Integral (PI) controller is used to allow the $\mathbf{B}$ point to track $r(k)$. The PI control's law can be written as in (2), resulting in the following state-space representation

$$\begin{aligned} x_c(k+1) &= x_c(k) + T_s(r(k) - y(k)) \\ u(k) &= K_i x_c(k) + K_p(r(k) - y(k)) \end{aligned} \tag{15}$$

where $x_c \in \mathbb{R}^2$, $K_p = k_p I_2 \in \mathbb{R}^{2\times 2}$ is the controller integral state vector, $T_s = t_s I_2 \in \mathbb{R}^{2\times 2}$, and $K_i = k_i I_2 \in \mathbb{R}^{2\times 2}$, $k_p, k_i \in \mathbb{R}$ are the controller's gains. Equivalently, (15) can be represented as the following input-output linear relation

$$\begin{bmatrix} x_c(k+1) \\ u(k) \end{bmatrix} = \underbrace{\begin{bmatrix} I_2 & T_s & -T_s \\ K_i & K_p & -K_p \end{bmatrix}}_{K} \begin{bmatrix} x_c(k) \\ r(k) \\ y(k) \end{bmatrix}. \tag{16}$$

To allow the control law (16) to be executed in encrypted form on the cloud, the matrix $K \in \mathbb{R}^{4\times 6}$ is there pre-uploaded and available in plaintext (non-encrypted matrix). On the other hand, the vector $v(k) = \begin{bmatrix} x_c(k) & r(k) & y(k) \end{bmatrix}^T \in \mathbb{R}^6$, before transmission to the cloud, is encoded. Consequently, the control law (16) is computed on the cloud, in an encrypted form, as:

$$\begin{aligned} \mathtt{E}[x_{c,l}(k+1)] &= (V_{l,1} \otimes \mathtt{E}[v_1(k)]) \oplus \ldots \oplus (V_{l,6} \otimes \mathtt{E}[v_6(k)]) \\ \mathtt{E}[u_l(k)] &= (Z_{l,1} \otimes \mathtt{E}[v_1(k)]) \oplus \ldots \oplus (Z_{l,6} \otimes \mathtt{E}[v_6(k)]), \\ & \qquad l = 1, 2, \end{aligned} \tag{17}$$

where $V = \begin{bmatrix} I_2 & T_s & -T_s \end{bmatrix} \in \mathbb{R}^{2\times 6}$, $Z = \begin{bmatrix} K_i & K_p & -K_p \end{bmatrix} \in \mathbb{R}^{2\times 6}$. On the plant's side, after decryption, i.e., $u(k) = [\mathtt{D}[\mathtt{E}[u_1(k)]], \mathtt{D}[\mathtt{E}[u_2(k)]]]^T$, the angular velocities of right and left wheels are recovered as

$$u^D(k) = H^{-1} T_{FL}(\theta(k)) u(k). \tag{18}$$

D. Setup and experimental results

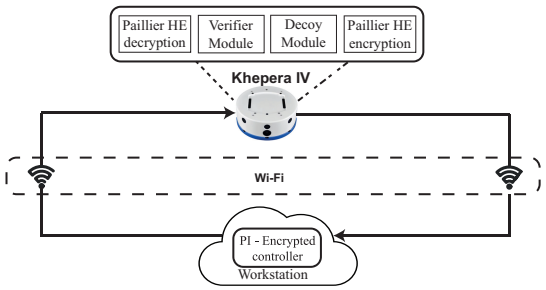


Fig. 3: Experimental setup.

The used experimental networked control system setup is illustrated in Fig. 3, where a Khepera IV robot is remotely and wirelessly controlled by a workstation acting as the remote cloud. The Khepera IV is a differential-drive robot with $R = 0.0210[m]$ and $D = 0.10470[m]$, and it is equipped with an 800MHz ARM Cortex-A8 Processor with C64x Fixed Point DSP core and 256 MB of RAM. The cloud is simulated using a workstation running Windows 10 with an Intel Core i9-13900KF processor with 24 cores, where the robot's encrypted control law (17) is executed. The Verifier, Decoder, Encryption and Decryption modules and (18) are all executed on the Khepera robot's. The remote exchange of encrypted data between the robot and the workstation has been implemented using the 802.11 b/g WiFi protocol has been used. The encrypted control law has been implemented using Phyton and the *eclib* library (`https://pypi.org/project/eclib/`) and using as control knobs $t_s = 0.15$[sec], $b = 0.1$, $k_p = 4$, and $k_i = 0.2$. Moreover, the Paillier encryption is performed using $|p| = |q| = 512$ and a quantization parameter $\delta = 0.0001$. To speed-up the arithmetic operations necessary to perform encryption, decryption and homomorphic multiplications, the C-coded Python extension module *gmpy2* (`https://pypi.org/project/gmpy2/`) has been used. The reference trajectory (14), whose path in the dashed red line shown in Fig. 4b, has been obtained interpolating a set of 22 waypoints distributed along the path using a cubic spline configured such that the average longitudinal velocity between any two consecutive waypoints is fixed and equal to $0.09[m/s]$.

The decoy module is configured with $n_d = 1$, $N_d = 2$, i.e., to randomly choose a single decoy from a pool of 2 pre-computed decoy tuples:

$$\begin{aligned} \mathcal{D} = \{ & (2 \cdot \mathbf{1}_2, 0.075 \cdot \mathbf{1}_2, \underbrace{\{\mathbf{0}_2, 2 \cdot \mathbf{1}_2, 2.5 \cdot \mathbf{1}_2\}}_{\text{decoy 1}}), \\ & (-3 \cdot \mathbf{1}_2, 4.85 \cdot \mathbf{1}_2, \underbrace{\{5 \cdot \mathbf{1}_2, \mathbf{1}_2, \mathbf{0}_2\}}_{\text{decoy 2}})\}. \end{aligned} \tag{19}$$

Consequently, for the given choices, the probability that any attack remains undetected over the discrete time interval $[0, k]$ is $(\frac{1}{2})^{k+1}$. At each k, the cloud receives

$$\begin{aligned} \mathtt{E}[X_c(k)] &= \left[\mathtt{E}[x_c(k)], \mathtt{E}[x_{c_d}^1(k)]\right] \Omega_k \\ \mathtt{E}[Y(k)] &= \left[\mathtt{E}[y(k)], \mathtt{E}[y_d^1(k)]\right] \Omega_k \\ \mathtt{E}[R(k)] &= \left[\mathtt{E}[r(k)], \mathtt{E}[r_d^1(k)]\right] \Omega_k \end{aligned}$$

and without the knowledge of Ω_k, it executes the encrypted PI control law (17) starting for each column of $\mathtt{E}[X_c(k)], \mathtt{E}[Y(k)], \mathtt{E}[R(k)]$. Then, all the encrypted outputs are collected into the following vectors

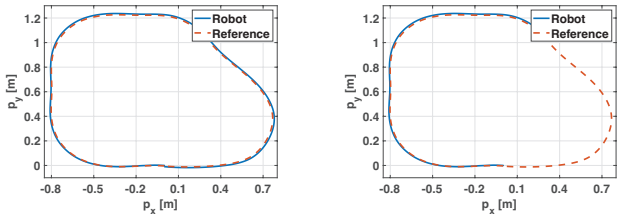
$$\begin{aligned} \mathtt{E}[U(k)] &= [\mathtt{E}[u(k)], \mathtt{E}[u_d^1(k)]] \Omega_k \\ \mathtt{E}[X_c(k+1)] &= [\mathtt{E}[x_c(k+1)], \mathtt{E}[x_{c_d}^1(k+1)]] \Omega_k \end{aligned}$$

and sent to the robot's Paillier HE decryptor module, which decrypts and sends the plaintext vectors to the Verifier. If at the time instant $\bar{k} \geq 0$, the Verifier finds an incorrect computation related to the decoy, it will stop the robot in the current position, i.e., $u^D(k) = [0, 0]^T, \forall\, k \geq \bar{k}$.

Two experiments have been performed to verify that the proposed verifiable computing protocol *(i)* does not interfere with tracking operations of the networked control system, *(ii)* and it is able to promptly detect cyber-attacks affecting the integrity of the encrypted control logic. In particular, in the first experiment, we consider a scenario where no cyber-attacks affect the encrypted computation, while in the second, a cyber-attack violates the integrity of the PI Encrypted controller starting from $k = 200\,(t = 30\,[\text{sec}])$ to disrupt the reference tracking task. We have configured the attacker to disrupt the encrypted PI logic as follows. First, it allows the computation of (17) for the first set of measurements and records the obtained encrypted outputs. Then, instead of re-computing (17) for the second set of data, it forces the output to be equal to the outcome of the first computation. As a consequence, $\forall\, k \geq 200$, the transmitted values are (with an equal probability of 0.5) *either*

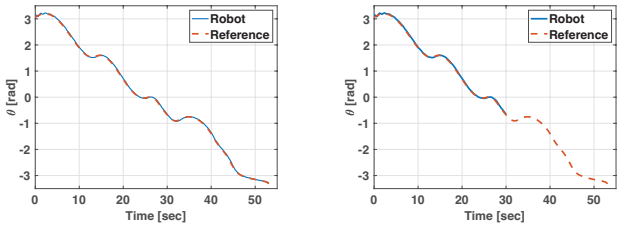
- (*Case 1*): $\mathtt{E}[U(k)] = [\mathtt{E}[u_d^1(k)], \mathtt{E}[u_d^1(k)]]$, $\mathtt{E}[X_c(k+1)] = [\mathtt{E}[x_{c_d}^1(k+1)], \mathtt{E}[x_{c_d}^1(k+1)]]$, *or*
- (*Case 2*): $\mathtt{E}[U(k)] = [\mathtt{E}[u(k)], \mathtt{E}[u(k)]]$, $\mathtt{E}[X_c(k+1)] = [\mathtt{E}[x_c^j(k+1)], \mathtt{E}[x_c^j(k+1)]]$,

where only *Case 1* bypasses the verification performed on the decoy. Note that even if the attack is aware of the used decoy set $\mathcal{D}$, the randomized nature of the used cryptosystem does not allow it to distinguish between real and decoy measurements. Consequently, the considered attack scenario is representative of any other deception attack against the integrity of the encrypted controller.



(a) Without attack. (b) With attacks.

Fig. 4: Robots trajectory.



(a) Without attacks. (b) With attacks.

Fig. 5: Robot's orientation.

The experimental results are collected in Figs. 4-6. In the subplots 4a and 5a it is possible to observe that, as expected, the proposed verifiable computing operations do not interfere with the real-time operations of the control loop. Indeed, as shown in the box plot of Fig. (6), the maximum time to execute the control loop of Fig. 3 is equal to $0.0631[\text{sec}]$ which is smaller than the used sampling time. Moreover, the computational overhead caused by the proposed scheme, mainly due to the encryption and decryption of the decoys, is smaller than $0.005\,[\text{sec}]$ with an average of $0.0032\,[\text{sec}]$. On the other hand, the subplots 4b and 5b show that the Verifier module was able to detect the attack at $t = 30[\text{sec}]$ and to stop the robot after such occurrence. The instantaneous detection finds justification in the fact that at $t = 30[\text{sec}]$, *decoy 1* was sent along the real measurement vectors while the cloud returned $U(30) = [[0.045, 0.082]^T], [0.045, 0.082]^T]$ (*Case 2* for the attacker's strategy) which was not compatible with the expected decoy output. The demo pertaining to the performed experiments is available at the following YouTube link: http://tinyurl.com/4a8u4y9r.

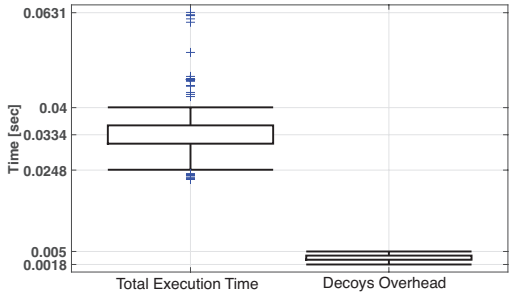


Fig. 6: Boxplot for encrypted control loop execution time.

V. Conclusions

We presented an efficient verifiable computing solution for encrypted cloud-based control systems. The proposed approach relies on probabilistically checkable proof that enables the plant's actuator to authenticate the computations performed by the encrypted networked controller without compromising the performance of the control scheme. The effectiveness and real-time applicability of the proposed scheme have been demonstrated through experiments using a remotely controlled Khepera-IV differential-drive robot.

References

[1] Y. Xia, Y. Zhang, L. Dai, Y. Zhan, and Z. Guo, "A brief survey on recent advances in cloud control systems," *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol. 69, no. 7, pp. 3108–3114, 2022.

[2] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, "Attack models and scenarios for networked control systems," in *Int. Conf. on High Confidence Networked Systems*, 2012, pp. 55–64.

[3] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[4] J. Kim, C. Lee, H. Shim, J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 175–180, 2016.

[5] N. Schlüter, P. Binfet, and M. S. Darup, "A brief survey on encrypted control: From the first to the second generation and beyond," *Annual Reviews in Control*, p. 100913, 2023.

[6] 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.

[7] A. M. Naseri, W. Lucia, M. Mannan, and A. Youssef, "On securing cloud-hosted cyber-physical systems using trusted execution environments," in *IEEE Int. Conf. on Autonomous Systems*, 2021, pp. 1–5.

[8] C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur, "Geppetto: Versatile verifiable computation," in *IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 253–270.

[9] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," *Communications of the ACM*, vol. 59, no. 2, pp. 103–112, 2016.

[10] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *IEEE Symposium on Security and Privacy*. IEEE, 2018, pp. 315–334.

[11] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *ACM SIGSAC Conf. on Computer and Communications Security*, 2014, pp. 844–855.

[12] D. Fiore, A. Nitulescu, and D. Pointcheval, "Boosting verifiable computation on encrypted data," in *Public-Key Cryptography: IACR International Conference on Practice and Theory of Public-Key Cryptography*. Springer, 2020, pp. 124–154.

[13] J. H. Cheon, D. Kim, J. Kim, S. Lee, and H. Shim, "Authenticated computation of control signal from dynamic controllers," in *IEEE Conf. on Decision and Control*. IEEE, 2020, pp. 3249–3254.

[14] R. Mahfouzi, A. Aminifar, S. Samii, P. Eles, and Z. Peng, "Secure cloud control using verifiable computation," in *IEEE Int. Conference on Omni-Layer Intelligent Systems*. IEEE, 2021, pp. 1–6.

[15] C. Crépeau, *Cut-and-choose protocol*. Boston, MA: Springer US, 2005, pp. 123–124. [Online]. Available: https://doi.org/10.1007/0-387-23483-7_92

[16] D. Chaum and et al., "Scantegrity ii: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes," *EVT*, vol. 8, no. 1, p. 13, 2008.

[17] N. P. Smart, *Cryptography made simple*. Springer, 2016.

[18] C. Escudero, C. Murguia, P. Massioni, and E. Zamaï, "Safety-preserving filters against stealthy sensor and actuator attacks," in *IEEE Conf. on Decision and Control (CDC)*. IEEE, 2023, pp. 5097–5104.

[19] C. Tiriolo, G. Franzè, and W. Lucia, "A receding horizon trajectory tracking strategy for input-constrained differential-drive robots via feedback linearization," *IEEE Trans. on Control Systems Technology*, vol. 31, no. 3, pp. 1460–1467, 2022.

[20] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Int. Conf. on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.