2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Towards Trustworthy AI: Sandboxing AI-based Unverified Controllers for Safe and Secure Cyber-Physical Systems

Bingzhuo Zhong[1], Siyuan Liu[2], Marco Caccamo[1], and Majid Zamani[3]

***Abstract*— In the past decade, artificial-intelligence-based (AI-based) techniques have been widely applied to design controllers over cyber-physical systems (CPSs) for complex control missions (e.g., motion planning in robotics). Nevertheless, AI-based controllers, particularly those developed based on deep neural networks, are typically very complex and are challenging to be formally verified. To cope with this issue, we propose a secure-by-construction architecture, namely Safe-Sec-visor architecture, to sandbox AI-based unverified controllers. By applying this architecture, the overall safety and security of CPSs can be ensured simultaneously, while formal verification over the AI-based controllers is *not* required. Here, we consider invariance and opacity properties as the desired safety and security properties, respectively. Accordingly, by leveraging a notion of (augmented) control barrier functions, we design a supervisor to check the control inputs provided by the AI-based controller and decide whether to accept them. At the same time, a safety-security advisor runs in parallel and provides fallback control inputs whenever the AI-based controller is rejected for safety and security reasons. To show the effectiveness of our approaches, we apply them to a case study on a quadrotor controlled by an AI-based controller. Here, the initial state of the quadrotor contains secret information which should not be revealed while the safety of the quadrotor should be ensured.**

I. Introduction

The past decade has witnessed remarkable achievements in artificial intelligence (AI) in many domains, such as natural language processing and image recognition. In the near future, plenty of AI-based controllers are also expected to be deployed in modern cyber-physical systems (CPSs) to accomplish complex control missions; typical scenarios include autonomous driving vehicles and smart buildings [1]. Nevertheless, verification of many AI-based controllers, particularly those developed based on deep neural networks, is a challenging task that is shown to be nondeterministic polynomial-time complete (NP-complete) in general [2]. Meanwhile, modern CPSs are typically safety-critical [3] and prone to various security threats [4]–[6] due to the tight interaction and information exchange between their cyber and physical components. Therefore, lack of verifying AI-based controllers might lead to disastrous consequences in real-life CPSs in terms of safety and security concerns.

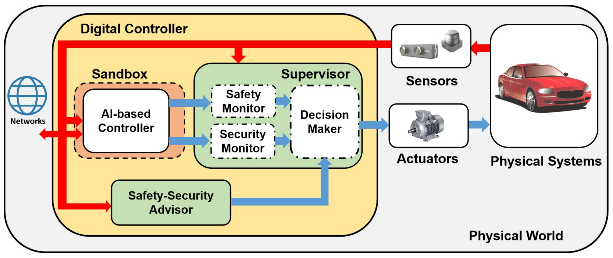

Fig. 1. Safe-Sec-visor architecture for sandboxing AI-based controllers concerning both safety and security properties.

In this paper, we focus on those CPSs in which AI-based controllers are deployed. In particular, we aim at providing formal guarantees regarding safety and security *simultaneously* over those CPSs without formally verifying the AI-based controllers. Concretely, inspired by the results in [7], [8], we propose a new architecture utilizing the idea of *sandbox* [9], namely *Safe-Sec-visor architecture* (see Fig. 1), for sandboxing AI-based unverified controllers. In particular, the proposed architecture consists of a safety-security advisor, and a supervisor that contains a safety monitor and a security monitor. At run-time, the supervisor decides whether to deploy the AI-based controller to control the system based on the decision of the safety and security monitors. The safety monitor rejects the AI-based controller whenever it endangers the overall safety of the system. Similarly, the security monitor is in charge of rejecting those control inputs from the AI-based controllers that would result in a violation of the desired security properties. In case the AI-based controller is rejected, the safety-security advisor is responsible for ensuring the overall safety and security of the system. Note that the safety-security advisor is supposed to be deployed as less as possible since it *only* focuses on keeping the system safe and secure, and we, therefore, need to exploit the functionalities offered by the AI-based controller.

Our contribution. To the best of our knowledge, this is the first work that introduces an architecture for sandboxing AI-based unverified controllers to provide formal guarantees regarding safety and security properties *simultaneously* over control systems with *continuous* state and input sets. Here, we would also like to mention those existing results (e.g. [10], [11]) in which formal guarantees are achieved by appropriately incorporating the desired objectives in the reward functions when training AI-based controllers. Com-

This work was supported in part by German Research Foundation (DFG) under grant ZA 873/7-1, by NSF under Grant ECCS-2015403, and the funding from Digital Futures, Sweden. Marco Caccamo was supported by an Alexander von Humboldt Professorship endowed by the German Federal Ministry of Education and Research.

[1]B. Zhong and M. Caccamo are with the TUM School of Engineering and Design, Technical University of Munich, Germany. Email: {bingzhuo.zhong,mcaccamo}@tum.de
[2]S. Liu is with the Division of Decision and Control Systems, KTH Royal Institute of Technology, Stockholm, Sweden. Email: siyliu@kth.se
[3]M. Zamani is with the Department of Computer Science, University of Colorado Boulder, USA. Email: majid.zamani@colorado.edu

pared with these results, the desired safety and security properties are decoupled from the construction of the AI-based controller in our proposed architecture. Therefore, our architecture can provide formal guarantees for any type of AI-based controllers regardless of their design.

Related work. To synthesize controllers enforcing safety properties while providing formal guarantees, discretization-based techniques (e.g., [12]–[14]) and discretization-free approaches (e.g., [15], [16]) have been developed in several results. The analysis of various security properties for CPSs has also attracted considerable attention in the literature [4], [5]. Particularly, for security properties formulated as various *opacity* notions, there have been some results dealing with their formal analysis over CPSs [17]–[19]. In [20], [21], a discretization-free scheme was introduced to synthesize secure-by-construction controllers which ensure safety and security properties simultaneously. Note that the results mentioned above focus on synthesizing controllers *directly* enforcing desired safety and security properties without inter-playing with any AI-based unverified controllers.

In the context of providing safety guarantees over control systems while deploying AI-based unverified controllers, some system-level, correct-by-construction schemes [22] have been proposed without requiring formal verification over those AI-based controllers. For example, shields [23] can be used to correct erroneous control inputs at run-time to ensure safety over discrete systems (e.g., [24]) and continuous-space systems (e.g., [25]). Reachability analysis-based techniques [26], [27] can also be leveraged to provide safety guarantees by checking the intersection between the unsafe and reachable sets of the systems. Alternatively, *Simplex* architecture (e.g. [28]) deploys a Lyapunov-function-based recovery region to achieve safety guarantees by regulating the unverified controllers. Later, the results in [29] deploy reachability analysis to enlarge the recovery region; [7] introduces a new architecture, namely *Safe-visor* architecture, to sandbox AI-based unverified controller and ensure the overall safety of CPSs. The results in [8], [30] further improve those in [7] by allowing more complex properties and enlarging the modeling framework. Note that all results above only focus on ensuring safety over control systems while deploying AI-based controllers.

II. Preliminary and Problem Formulation

A. Notation and Preliminary

We denote by $\mathbb{R}$ and $\mathbb{N}$ the set of real numbers and non-negative integers, respectively. These symbols are annotated with subscripts to restrict them in a usual way, e.g., $\mathbb{R}_{>0}$ denotes the set of positive real numbers. For $a, b \in \mathbb{R}$ (resp. $a, b \in \mathbb{N}$) with $a \leq b$, the closed, open and half-open intervals in $\mathbb{R}$ (resp. $\mathbb{N}$) are denoted by $[a, b]$, (a, b), $[a, b)$, and $(a, b]$, respectively. Given $N \in \mathbb{N}_{\geq 1}$ vectors $x_i \in \mathbb{R}^{n_i}$, with $i \in [1; N]$, $n_i \in \mathbb{N}_{\geq 1}$, and $n = \sum_i n_i$, we denote the concatenated vector in $\mathbb{R}^n$ by $x = [x_1; \ldots; x_N]$ and the Euclidean norm of x by $\|x\|$. Given a set $Y \subseteq \mathbb{R}^{2n}$, we denote by $\overline{\mathbf{Proj}}(Y)$ and $\underline{\mathbf{Proj}}(Y)$ the projection of the set Y on to the first and the last n coordinates, respectively, i.e., $\overline{\mathbf{Proj}}(Y) := \{y \in \mathbb{R}^n | \exists \hat{y} \in \mathbb{R}^n, \text{s.t. } [y; \hat{y}] \in Y\}$, and $\underline{\mathbf{Proj}}(Y) := \{\hat{y} \in \mathbb{R}^n | \exists y \in \mathbb{R}^n, \text{s.t. } [y; \hat{y}] \in Y\}$. Given a matrix A, we denote by $A^\top$ and $\{A\}_{i,j}$ the transpose and the entry in i-th row and j-th column of A, respectively. Additionally, $\mathbf{0}$ represents zero matrices of appropriate dimensions. Given sets X and Y, the complement of X with respect to Y is defined as $Y \backslash X = \{x \in Y \mid x \notin X\}$. The Cartesian product of two sets X and Y is defined as $X \times Y = \{(x, y) \mid x \in X, y \in Y\}$. Given functions $f : X \to Y$ and $g : A \to B$, we define $f \times g : X \times A \to Y \times B$.

Additionally, we need the following definitions throughout the paper, which are borrowed from [31, Section 3].

Definition 2.1: Consider $x := [x_1; \ldots; x_n] \in \mathbb{R}^n$. A *monomial* $m : \mathbb{R}^n \to \mathbb{R}$ in x is a function defined as $m(x) := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, with $\alpha_1, \ldots, \alpha_n \in \mathbb{N}$, and $\mathcal{M}(x)$ denotes the sets of all monomials over $x \in \mathbb{R}^n$. A function $M : \mathbb{R}^n \to \mathbb{R}^{r_1 \times r_2}$ is a *matrix monomial* if $\{M(x)\}_{i,j} \in \mathcal{M}(x)$, $\forall i \in [1, r_1], \forall j \in [1, r_2]$. We denote by $\mathcal{M}^{\mathsf{m}}(x)$ the set of all matrix monomials over $x \in \mathbb{R}^n$.

Definition 2.2: A *polynomial* $h : \mathbb{R}^n \to \mathbb{R}$ is a sum of a finite number of monomials, as $p(x) := \sum_i^{N_\mathrm{i}} c_i m_i(x)$, with $c_i \in \mathbb{R}$, $m_i(x) \in \mathcal{M}(x)$. We denote by $\mathcal{P}(x)$ the set of polynomials over $x \in \mathbb{R}^n$. Moreover, a function $\mathbf{P} : \mathbb{R}^n \to \mathbb{R}^{r_1 \times r_2}$ is a *matrix polynomial* if $\{\mathbf{P}(x)\}_{i,j} \in \mathcal{P}(x)$, $\forall i \in [1, r_1], \forall j \in [1, r_2]$. We denote by $\mathcal{P}^{\mathsf{m}}(x)$ the set of all matrix polynomials over $x \in \mathbb{R}^n$.

B. System Model

In this paper, we are interested in a class of discrete-time control systems, which is defined below.

Definition 2.3: A *discrete-time control system* (dt-CS) Σ is a tuple $\Sigma := (X, X_0, U, f, Y, h)$, where $X \subseteq \mathbb{R}^n$ denotes the state set; $X_0 \subseteq X$ denotes the initial state set; $U \subset \mathbb{R}^m$ is the input set defined as

$$U := \{u \in \mathbb{R}^m | \rho_j(u) \leq 0, j \in [1, \mathrm{j}], \mathrm{j} \in \mathbb{N}\}, \qquad \text{(II.1)}$$

in which $\rho_j(u) \in \mathcal{P}(u)$ are some known polynomial functions, and $Y \subseteq \mathbb{R}^q$ denotes the output set. Moreover, the function $f : X \times U \to X$ is the state transition function, and $h : X \to Y$ is the output function.

Note that the input set as in (II.1) represents the input saturation of the physical system. Here, we focus on those dt-CS Σ which are polynomial affine [32], as described below:

$$\Sigma : \begin{cases} x(k+1) = & f(x(k), \nu(k)) \\ \qquad\quad := & A\mathcal{X}(x(k)) + B\mathcal{U}(x(k))\nu(k), \\ \quad y(k) = & h(x(k)), \qquad\quad k \in \mathbb{N}, \end{cases} \qquad \text{(II.2)}$$

where $x(k) \in X$, $\nu(k) \in U$, and $y(k) \in Y$, in which $A \in \mathbb{R}^{n \times N_x}$ and $B \in \mathbb{R}^{n \times N_u}$ are some constant matrices, $\mathcal{U}(x), \mathcal{X}(x) \in \mathcal{M}^{\mathsf{m}}(x)$, in which $\mathcal{X}(x(k)) := \mathcal{H}(x(k))x(k)$ with $\mathcal{H}(x(k)) \in \mathcal{M}^{\mathsf{m}}(x)$. Here, $\nu = (\nu(0), \ldots, \nu(k), \ldots)$ is an input run of Σ, and $\mathbf{x}_{x_0,\nu} := (x(0), \ldots, x(k), \ldots)$ represents a state run of Σ starting from initial state x_0 under input run ν, i.e., $x(0) = x_0$, $x(k+1) = f(x(k), \nu(k))$, $\forall k \in \mathbb{N}$.

C. Problem Formulation

In this paper, we aim at constructing a Safe-Sec-visor architecture as in Fig. 1, which allows the application of AI-based unverified controllers for controlling dt-CS, while guaranteeing the desired safety and security properties. Here, the desired safety properties require that

$$\forall x_0 \in X_0, \text{one has } \mathbf{x}_{x_0,\nu}(k) \notin X_d, \forall \nu, \forall k \in \mathbb{N}, \tag{II.3}$$

where $X_d \subseteq X$ is an *unsafe set*. In other words, any state-run of the system starting from X_0 should not reach X_d. In particular, we focus on sets X_d satisfying

$$X \backslash X_d := \{x \in X | a_i x \leq 1, i \in [1, \mathrm{i}]\}, \tag{II.4}$$

where $a_i^\top \in \mathbb{R}^n$ are some known constant vectors. In other words, the complement of unsafe set X_d, a.k.a. the safe region within X, can be described as a polytope.

The security property considered in this paper is an important class of information-flow security property called *opacity* [33]. Roughly speaking, opacity characterizes the system's plausible deniability for its secret behavior in the sense that its secret and nonsecret behaviors are indistinguishable in the eye of an outside observer (a.k.a. *intruder*). In this context, we assume that the intruder can observe the output sequences of the system with a certain measurement precision. The intruder is also assumed to know the system model and its dynamics. By observing the output sequences and without actively affecting the behavior of the system, the intruder tries to infer certain secret information from the system based on the knowledge of the system model. In this paper, we are interested in a state-based notion of opacity called *approximate initial-state opacity* [17], which is used to model security requirements in many applications, including secure cryptographic protocols and tracking problems in sensor networks [34]. Here, we denote by $X_s \subset X$ the set of secret states, and the formal definition of approximate initial-state opacity is recalled from [17] as follows.

Definition 2.4: Consider a dt-CS $\Sigma = (X, X_0, U, f, Y, h)$, a set of secret states $X_s \subset X$, and a constant $\delta \in \mathbb{R}_{\geq 0}$. System Σ is said to be *δ-approximate initial-state opaque* if for any $x_0 \in X_0 \cap X_s$ and any finite state run $\mathbf{x}_{x_0,\nu} = (x_0, \dots, x_T)$, there exists a finite state run $\mathbf{x}_{\hat{x}_0,\hat{\nu}} = (\hat{x}_0, \dots, \hat{x}_T)$, with $\hat{x}_0 \in X_0 \backslash X_s$, such that

$$\|h(x_i) - h(\hat{x}_i)\| \leq \delta, \forall i \in [0, T]. \tag{II.5}$$

As a key insight, δ-approximate initial-state opacity requires that an outside intruder is never certain whether the system was initiated from a secret state, in which δ models the precision of the intruder's observation in the sense that given $\hat{y} := (\hat{y}_0, \dots, \hat{y}_T)$ the observation of the intruder, one has

$$\|y_i - \hat{y}_i\| \leq \delta, \forall i \in [0, T]. \tag{II.6}$$

where $y := (y_0, \dots, y_T)$ is the actual output sequence of the system corresoponding to $\hat{y}$. It is also worth noting that to ensure the desired approximate initial-state opacity over Σ, the secret of the system should at least not be revealed initially; otherwise, the desired opacity is trivially violated. Therefore, we assume, without loss of generality,

$$\forall x_0 \in X_0 \cap X_s, \{x \in X_0 | \|h(x) - h(x_0)\| \leq \delta\} \nsubseteq X_s. \tag{II.7}$$

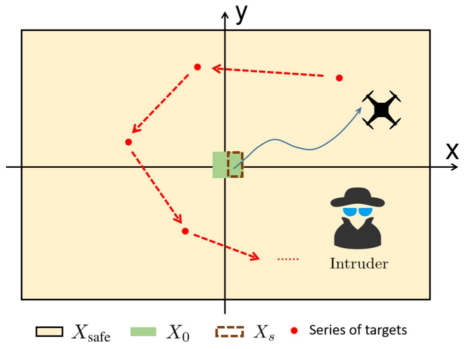

Fig. 2. A quadrotor tracks a series of targets. Meanwhile, the positions of the quadrotor are observed by a malicious intruder remotely.

For a compact description of the system and the desired safety and security properties, in the remaining discussion, we incorporate X_d and X_s in the system definition and use

$$\Sigma := (X, X_0, X_s, X_d, U, f, Y, h), \tag{II.8}$$

to denote a dt-CS as in Definition 2.3 with the desired safety and security properties. Now, we are ready to formulate the main problem to be tackled in this paper.

Problem 2.5: Given a dt-CS $\Sigma := (X, X_0, X_s, X_d, U, f, Y, h)$ in (II.8), and a constant $\delta \in \mathbb{R}_{\geq 0}$, design a Safe-Sec-visor architecture as in Fig. 1 (if existing) to check the validity of the control inputs provided by the AI-based controller such that 1) and 2) hold:

1) (Safety) Σ is safe such that (II.3) holds;
2) (Opacity) Σ is δ-approximate initial-state opaque.

To better illustrate the motivation and the theoretical results in this paper, we deploy the following running example, which will also be the case study of the paper.

Example. Here, we consider an example in which a quadrotor tracks a series of changing targets on a 2-dimensional plane (x-y plane), as shown in Fig. 2. Meanwhile, the positions of the quadrotor are observed by a malicious intruder, with an observation precision $\delta = 2$ as described in (II.6). Here, it is desired to not reveal to the intruder whether the quadrotor starts from the secret region X_s. At the same time, due to air traffic regulations, the quadrotor is required to stay within a safety region X_{safe}.

By leveraging the feedback linearization technique in [35], the quadrotor can be modeled by

$$\Sigma : \begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k), \quad k \in \mathbb{N}, \end{cases} \tag{II.9}$$

with

$$A := \begin{bmatrix}1 & \Delta t & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & \Delta t\\ 0 & 0 & 0 & 1\end{bmatrix}, B := \begin{bmatrix}\Delta t^2/2\\ \Delta t\\ \Delta t^2/2\\ \Delta t\end{bmatrix}, C := \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}.$$

Here, $\Delta t = 0.1$s is the sampling time; $x := [x_{\mathsf{x}}; v_{\mathsf{x}}; x_{\mathsf{y}}; v_{\mathsf{y}}]$ and $u := [u_{\mathsf{x}}; u_{\mathsf{y}}]$ denote the state and the control input of the quadrotor, respectively, with x_{i}, v_{i}, and u_{i} being the position, velocity, and acceleration of the drone on the i axis, $\mathsf{i} \in \{\mathsf{x}, \mathsf{y}\}$, respectively; y is the output of the system, which can be observed by a malicious intruder remotely. Here, we consider the state set $X := [-120, 120] \times [-5, 5] \times [-120, 120] \times [-5, 5]$, initial set $X_0 := \big([-0.5, 0] \times \{0\} \times [-0.05, 0.05] \times \{0\}\big) \cup \big([0, 0.5] \times [-0.85, 0.85] \times [-0.3, 0.3] \times [-0.85, 0.85]\big)$, secret set $X_s := [0, 0.5] \times [-0.85, 0.85] \times [-0.3, 0.3] \times [-0.85, 0.85]$, and unsafe set $X_d := X \setminus \big([-100, 100] \times [-3, 3] \times [-100, 100] \times [-3, 3]\big)$ (note that $X_d := X \setminus X_{\text{safe}}$). In brief, the position of the quadrotor $(x_{\mathsf{x}}, x_{\mathsf{y}})$ is required to be within the region $[-100, 100] \times [-100, 100]$, while the velocity of the quadrotor cannot exceed 3 m/s on both axes. Moreover, we consider $u_{\mathsf{i}} \in [-2, 2]$ m/s^2, with $\mathsf{i} \in \{\mathsf{x}, \mathsf{y}\}$ as input constraints. $\diamond$

III. Construction of Safe-Sec-visor Architecture

In this section, we focus on constructing a Safe-Sec-visor architecture as in Fig. 1 to solve Problem 2.5. To this end, we first propose in Section III-A a notion of *(augmented) control barrier functions* [20], [21], which is used to construct the Safe-Sec-visor architecture in Section III-B.

A. (Augmented) Control Barrier Functions

To introduce notion of (augmented) control barrier functions, we need to define an *augmented system* associated with Σ, which is formulated as below.

Definition 3.1: Consider a dt-CS $\Sigma = (X, X_0, X_s, X_d, U, f, Y, h)$ as in (II.8). An *augmented system* associated with Σ is the product between Σ and itself and is defined as a tuple

$$\Sigma \times \Sigma := (X \times X, X_0 \times X_0, X_s \times X_s, X_d \times X_d, U \times U, f \times f, Y \times Y, h \times h). \quad \text{(III.1)}$$

Here, $(x, \hat{x}) \in X \times X$ is a state pair of $\Sigma \times \Sigma$, and $(\mathbf{x}_{x_0,\nu}, \mathbf{x}_{\hat{x}_0,\hat{\nu}})$ denotes the state trajectory of $\Sigma \times \Sigma$ starting from $(x_0, \hat{x}_0)$ under input run $(\nu, \hat{\nu})$.

Moreover, some conditions introduced in the next assumption are required for proposing the notion of (augmented) control barrier functions.

Assumption 3.2: Consider a dt-CS $\Sigma := (X, X_0, X_s, X_d, U, f, Y, h)$ as in (II.8), its associated augmented system $\Sigma \times \Sigma$ as in Definition 3.1, the polynomial-type state transition function as in (II.2), and set X_d as in (II.4). Given a set

$$\bar{\mathcal{R}} := \{(x, \hat{x}) \in X \times X | b_t[x; \hat{x}] \leq 1, t \in [1, \mathsf{t}]\} \subseteq \{(x, \hat{x}) \in X \times X |\ \|h(x) - h(\hat{x})\| \leq \delta\}, \quad \text{(III.2)}$$

with $b_t^\top \in \mathbb{R}^{2n}$ being some given constant vectors, we assume that the following conditions hold:

$$\begin{bmatrix} Q & \mathsf{g}(x)^\top \\ \mathsf{g}(x) & Q \end{bmatrix} \succeq 0, \forall x \in \mathbb{R}^n, \quad \text{(III.3)}$$

$$a_i Q a_i^\top \leq 1,\ i \in [1, \mathsf{i}]; \quad \text{(III.4)}$$

$$\begin{bmatrix} Q_o & \mathsf{g}_o(x, \hat{x})^\top \\ \mathsf{g}_o(x, \hat{x}) & Q_o \end{bmatrix} \succeq 0, \forall (x, \hat{x}) \in \mathbb{R}^{2n}, \quad \text{(III.5)}$$

$$b_t Q_o b_t^\top \leq 1,\ t \in [1, \mathsf{t}]; \quad \text{(III.6)}$$

for some $Q \in \mathbb{R}^{n \times n}$, $Q_o \in \mathbb{R}^{2n \times 2n}$, $\bar{K}(x) \in \mathcal{P}^{\mathsf{m}}(x)$, and $\bar{K}_o(x, \hat{x}) \in \mathcal{P}^{\mathsf{m}}(x, \hat{x})$, in which $\mathsf{g}(x) := A\mathcal{H}(x)Q + B\mathcal{U}(x)\bar{K}(x)$, $\mathsf{g}_o(x, \hat{x}) := A_o(x, \hat{x})Q_o + B_o(\hat{x})\bar{K}'_o(x, \hat{x})$, with

$$A_o := \begin{bmatrix} A\mathcal{H}(x) + B\mathcal{U}(x)K(x) & \mathbf{0} \\ \mathbf{0} & A\mathcal{H}(\hat{x}) \end{bmatrix}, B_o := \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & B\mathcal{U}(\hat{x}) \end{bmatrix},$$

$\bar{K}'_o(x, \hat{x}) := [\mathbf{0}; \bar{K}_o(x, \hat{x})]$, and $K(x) := \bar{K}(x)Q^{-1}$. Note that one can readily check Assumption 3.2 by checking the feasibility of conditions (III.3)-(III.6) using semi-definite-programming (SDP) solver (e.g., `Mosek` [36]). With Definitions 3.1 and Assumption 3.2, we propose a notion of (augmented) control barrier functions [20] that will be used to design the Safe-Sec-visor architecture in Section III-B.

Definition 3.3: (*(Augmented) Control Barrier Functions*) Consider a dt-CS $\Sigma := (X, X_0, X_s, X_d, U, f, Y, h)$ as in (II.8) and its associated augmented system $\Sigma \times \Sigma$ as in Definition 3.1. Suppose that there exists positive-definite matrices $Q \in \mathbb{R}^{n \times n}$, $Q_o \in \mathbb{R}^{2n \times 2n}$, and matrix polynomials $\bar{K}(x) \in \mathcal{P}^{\mathsf{m}}(x), \bar{K}_o(x, \hat{x}) \in \mathcal{P}^{\mathsf{m}}(x, \hat{x})$ such that conditions (III.3)-(III.6) in Assumption 3.2 hold. Then, functions

$$\mathcal{B} := x^\top Q^{-1} x - c_1,\ \mathcal{B}_O := [x; \hat{x}]^\top Q_o^{-1} [x; \hat{x}] - c_2 \quad \text{(III.7)}$$

are called *control barrier functions* (CBF) for Σ and *augmented control barrier functions* (ACBF) for the augmented systems $\Sigma \times \Sigma$, respectively, for some $c_1, c_2 \in (0, 1]$, if the following conditions hold:

- **(cd.1)** $\forall x \in \mathcal{S}$, one has $u := \bar{K}(x)Q^{-1}x \in U$;
- **(cd.2)** $X_0 \subset \mathcal{S}$;
- **(cd.3)** $\forall (x, \hat{x}) \in \mathcal{S}_O$, $\hat{u} := \bar{K}_o(x, \hat{x})Q_o^{-1}[x; \hat{x}] \in U$;
- **(cd.4)** $\exists \mathcal{R}_0 \subset X \times X$, such that $\emptyset \neq \mathcal{R}_0 \subset \mathcal{S}_O$, $\overline{\mathbf{Proj}}(\mathcal{R}') \subseteq \overline{\mathbf{Proj}}(\mathcal{R}_0)$, and $\underline{\mathbf{Proj}}(\mathcal{R}_0) \subseteq \underline{\mathbf{Proj}}(\mathcal{R}')$;

where $\mathcal{S} := \{x \in \mathbb{R}^n | x^\top Q^{-1} x - c_1 \leq 0\}$, $\mathcal{S}_O := \{(x, \hat{x}) \in \mathbb{R}^n \times \mathbb{R}^n | [x; \hat{x}]^\top Q_o^{-1} [x; \hat{x}] - c_2 \leq 0\}$, and $\mathcal{R}' := \{(x, \hat{x}) \in (X_0 \cap X_s) \times X_0 \setminus X_s | \|h(x) - h(\hat{x})\| \leq \delta - \epsilon\}$, with some $\epsilon \in [0, \delta]$.

Remark 3.4: As a key insight, the CBF $\mathcal{B}$ and the ACBF $\mathcal{B}_O$ play a key role in designing the safety and security monitors in Fig. 1, respectively (cf. Algorithm 1). In practice, given the state transition function as in (II.2), one can readily select a set $\bar{\mathcal{R}}$ as in (III.2) and compute Q, Q_o, $\bar{K}(x)$, and $\bar{K}_o(x, \hat{x})$ accordingly since conditions (III.3)-(III.6) formulate a sum-of-square programming problem [37] that can be solved by a SDP solver (e.g., `Mosek` [36]). Having Q, Q_o, $\bar{K}(x)$, and $\bar{K}_o(x, \hat{x})$ candidates, one can then proceed with checking whether or not **(cd.1)** -**(cd.4)** hold for some $c_1, c_2 \in (0, 1]$. $\diamond$

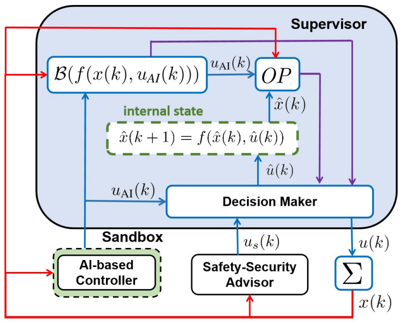

Fig. 3. Running mechanism of the Safe-Sec-visor architecture, with OP as in Algorithm 1, and $\mathcal{B}$ being the CBF as in (III.7).

It is also worth noting that in case (**cd.1**)-(**cd.4**) do not hold for the obtained Q, Q_o, $\bar{K}(x)$, and $\bar{K}_o(x,\hat{x})$, one can also leverage the conditions in [21, Theorem 3.3] and deploy an iteration scheme to expand the region specified by the sets $\mathcal{S}$ and $\mathcal{S}_O$. In this paper, we presume that one can find valid control barrier function $\mathcal{B}$ and augmented control barrier function $\mathcal{B}_O$ as in (III.7). Now, we focus on how to construct the Safe-Sec-visor architecture for Problem 2.5 by leveraging these functions, which is the main contribution of this paper and discussed in the following subsection.

Example (continued). Consider again the quadrotor tracking example as in Fig. 2. Given the model of the quadrotor as in (II.9), in order to compute the (augmented) control barrier functions as in Definition 3.3, we choose the set $\bar{\mathcal{R}}$ as in (III.2), with $b_1 = [0.711; 0; 0; 0; -0.711; 0; 0; 0]^\top$, $b_2 = [0; 0; 0.711; 0; 0; 0; -0.711; 0]^\top$, $b_3 = -b_1$, and $b_4 = -b_1$. Then, following the instruction in Remark 3.4, we deployed `Mosek` [36] and `YALMIP` [38] and obtained Q, Q_o, $\bar{K}(x)$, and $\bar{K}_o(x,\hat{x})$ as in (III.8)-(III.9) fulfilling conditions (III.3)-(III.6). Having Q, Q_o, $\bar{K}(x)$, and $\bar{K}_o(x,\hat{x})$, we verified that (**cd.1**)- (**cd.4**) in Definition 3.3 are satisfied, with $c_1 = c_2 = 1$, and the set $\mathcal{R}_0 = \{x \in X_s, \hat{x} \in [-0.01, 0) \times \{0\} \times [-0.01, 0.01) \times \{0\} \mid ||h(x){-}h(\hat{x})|| \leq \delta\}$. Therefore, one can construct the CBF and ACBF as in (III.7) using the obtained Q, Q_o, c_1, and c_2. $\diamond$

B. Design of Safe-Sec-visor Architecture

Here, the running mechanism of the Safe-Sec-visor architecture is formally proposed in Algorithm 1 and depicted in Fig. 3. In particular, lines 8-9, lines 11-14, and lines 19-20 correspond to the running mechanism of the safety monitor, the security monitor, and the safety-security advisor, respectively. Having Algorithm 1 in hand, we are ready to propose the main results of this paper, which provides a solution to Problem 2.5 with a formal guarantee using the Safe-Sec-visor architecture.

Theorem 3.5: Consider a dt-CS $\Sigma := (X, X_0, X_s, X_d, U, f, Y, h)$ as in (II.8). By applying the Safe-Sec-visor architecture as in Algorithm 1 for all $k \in \mathbb{N}$, one obtains that Σ is safe and δ-approximate initial-state opaque as stated in Problem 2.5 while running an AI-based unverified controller.

The proof of the Theorem 3.5 is provided in the Appendix. To demonstrate the effectiveness of the Safe-Sec-visor architecture for ensuring the desired security properties, we will show in the case study that without using our architecture, some state runs of the system indeed reveal the secret information of the system starting from the secret region (cd. Fig. 6). To this end, we will use the following results to identify whether or not a state run $\mathbf{x}_{x_0,\nu}$ of the system infers a violation of the desired δ-approximate initial-state opacity property.

Theorem 3.6: Consider a dt-CS $\Sigma := (X, X_0, X_s, X_d, U, f, Y, h)$ as in (II.8) and a state run $\mathbf{x}_{x_0,\nu} = (x_0, \ldots, x_T)$ generated by Σ. The state run $\mathbf{x}_{x_0,\nu}$ violates the desired δ-approximate initial-state opaque if $\exists k \in [0, T]$ such that $\min_{\hat{x} \in P(k)} ||h(x_k) - h(\hat{x})|| > \delta$, in which the set $P(k)$ is defined as

$$\begin{aligned} P(k) \supseteq \{\hat{x} \in \mathbb{R}^n | \exists \hat{x}_0 \in \underline{\mathbf{Proj}}(\mathcal{R}'), \\ \exists \hat{\nu} := [\hat{\nu}_1, \ldots, \hat{\nu}_k], \text{s.t. } \hat{x} = \mathbf{x}_{\hat{x}_0,\hat{\nu}}(k)\}, \end{aligned} \quad \text{(III.10)}$$

with the set $\mathcal{R}'$ being defined as in Definition 3.3.

The proof of the Theorem 3.6 is provided in the Appendix. Note that the set $P(k)$ is essentially (an over-approximation of) the k-step reachable set of the dt-CS Σ from the set $\underline{\mathbf{Proj}}(\mathcal{R}')$ alongside the state transition function f. One can use existing tools, such as `MPT3` [39] and `CORA` [40], to compute such sets.

IV. Simulation Results of the Case Study

In this section, we construct the Safe-Sec-visor architecture leveraging the CBF and ACBF obtained in Section III-A, and the results in Section III-B. Then, we apply this architecture to the quadrotor tracking example as in Fig. 2. Here, we deploy an AI-based controller which is supposed to enforce the quadrotor tracking a series of changing targets. Here, the AI-based controller contains a deep-neural-network-based (DNNs-based) agent, which is trained by leveraging DDPG algorithm [41] and works as a setpoint provider for low-level position controller. This agent takes the desired target, the current positions and velocities of the quadrotor as inputs and provides the position and velocity setpoints for the quadrotor. Note that we are not describing the details of how to train the agent here since designing and improving the performance of AI-based unverified controllers are out of the scope of this paper. The AI-based controller deployed here is only for demonstration purposes. In particular, our architecture can be deployed to any "off-the-shelf" AI-based controller regardless of their performance, while a formal guarantee for ensuring the desired safety and security properties can still be provided.

To simulate the system, we randomly selected 300 initial states $x(0)$ from the set $X_0 \cap X_s$ and simulated the system for 600 time steps. The simulation results are summarized in Table I and depicted in Figs. 4 and 5. When deploying the Safe-Sec-visor architecture, 30.78% of the control input provided by the AI-based controllers are accepted, while

$$Q = \begin{bmatrix} 9802.884 & -100.016 & 0 & 0 \\ -100.016 & 8.616 & 0 & 0 \\ 0 & 0 & 9802.884 & -100.016 \\ 0 & 0 & -100.0157 & 8.616 \end{bmatrix}, \bar{K}^\top(x) = \begin{bmatrix} -0.0004 & 0 \\ -0.0379 & 0 \\ 0 & -0.0004 \\ 0 & -0.0379 \end{bmatrix}, \tag{III.8}$$

$$Q_o = 10^5 \times \begin{bmatrix} 4.737 & -0.062 & 0 & 0 & 4.737 & -0.062 & 0 & 0 \\ -0.062 & 0.002 & 0 & 0 & -0.062 & 0.002 & 0 & 0 \\ 0 & 0 & 4.760 & -0.062 & 0 & 0 & 4.760 & -0.063 \\ 0 & 0 & -0.062 & 0.002 & 0 & 0 & -0.062 & 0.002 \\ 4.737 & -0.062 & 0 & 0 & 4.737 & -0.062 & 0 & 0 \\ -0.062 & 0.002 & 0 & 0 & -0.062 & 0.002 & 0 & 0 \\ 0 & 0 & 4.760 & -0.062 & 0 & 0 & 4.760 & -0.062 \\ 0 & 0 & -0.062 & 0.002 & 0 & 0 & -0.062 & 0.002 \end{bmatrix}, \bar{K}_o^\top(x,\hat{x}) = \begin{bmatrix} 1.510 & 0 \\ 0.817 & 0 \\ 0 & 1.510 \\ 0 & 0.817 \\ -1.511 & 0 \\ -0.908 & 0 \\ 0 & -1.511 \\ 0 & -0.908 \end{bmatrix}. \tag{III.9}$$

Algorithm 1: Running mechanism of the Safe-Sec-visor architecture

```
   Input: A dt-CS $\Sigma := (X, X_0, X_s, X_d, U, f, Y, h)$ as
          in (II.8) and its associated augmented system
          as in Definition 3.1; a CBF $\mathcal{B}$, an ACBF $\mathcal{B}_O$,
          matrix polynomials $\bar{K}(x)$ and $\bar{K}_o(x,\hat{x})$, and
          the set $\mathcal{R}_0$ satisfying (cd.4) in Definition 3.3;
          initial state $x_0$; and inputs $u_{\mathrm{AI}}(k)$ provided by
          the AI-based controller.
1  $k = 0$, $x(0) = x_0$
2  while True do
3      if $k = 0$ then
4          Initialize $\hat{x}(0) = \hat{x}_0$ such that $(x_0, \hat{x}_0) \in \mathcal{R}_0$.
5      else
6          Update the state $x(k)$ from $\Sigma$.
7      Update $u_{\mathrm{AI}}(k)$ from the AI-based controller.
8      if $\mathcal{B}(f(x(k), u_{AI}(k))) > 0$ then
9          Safety monitor rejects $u_{\mathrm{AI}}(k)$.
10     else
11         Solve the optimization problem OP:
               $\min_{\hat{u}\in U} ||h(f(x(k), u_{\mathrm{AI}}(k))) - h(f(\hat{x}(k), \hat{u}))||$
               s.t. $\mathcal{B}_O(f(x(k), u_{\mathrm{AI}}(k)), f(\hat{x}(k), \hat{u})) \le 0$
           if OP is feasible then
12             $u_{\mathrm{AI}}(k)$ is accepted.
13         else
14             Security monitor rejects $u_{\mathrm{AI}}(k)$.
15     if $u_{AI}(k)$ is accepted then
16         Update $u(k) := u_{\mathrm{AI}}(k)$ to control the dt-CS $\Sigma$
             at time instant $k$.
17         Update $\hat{x}(k+1) := f(\hat{x}(k), \hat{u}(k))$, with
             $\hat{u}(k) := \hat{u}$ obtained by solving OP in line 11.
18     else
19         Update $u(k) := u_s(k)$, with
             $u_s(k) = \bar{K}(x(k))Q^{-1}x(k)$ to control the
             dt-CS $\Sigma$ at time instant $k$.
20         Update $\hat{x}(k+1) := f(\hat{x}(k), \hat{u}(k))$, with
             $\hat{u}(k) := \bar{K}_o(x(k), \hat{x}(k))Q_o^{-1}[x(k); \hat{x}(k)]$.
21     $k = k + 1$.
   Output: $u(k)$ for controlling $\Sigma$ at each time step $k$.
```

	With Safe-Sec-visor architecture	Without Safe-Sec-visor architecture
Percentages of satisfying the safety properties	100%	0%
Percentages of satisfying the opacity properties	100%	71.66%

TABLE I

SIMULATION RESULTS OF THE CASE STUDY WITH AND WITHOUT USING THE SAFE-SEC-VISOR ARCHITECTURE.

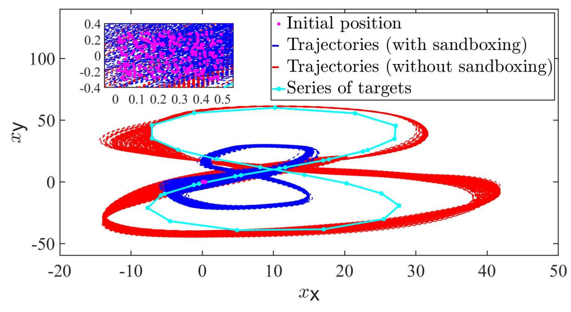

Fig. 4. Initial positions and trajectories of the quadrotor's positions with and without using the Safe-Sec-visor architecture.

the desired safety and initial-state opacity properties are satisfied. In contrast, when the Safe-Sec-visor architecture is not deployed to sandbox the AI-based controller, all the trajectories violate the desired safety properties. Additionally, 28.34% of the trajectories reveal the fact that the quadrotor starts from the secret region. Here, we demonstrate one such trajectory in Fig. 6 with the help of Theorem 3.6, in which the reachable sets are computed using `MPT3` [39].

V. Conclusion

In this paper, we proposed for the first time a secure-by-construction architecture, so-called *Safe-Sec-visor architecture*, to ensure safety and security properties simultaneously over cyber-physical systems. This architecture is inspired by the notion of *Safe-visor architecture* [7] and consists of 1) a safety monitor that identifies those control inputs from the AI-based controller endagering the overall safety of the system; 2) a security monitor that rejects the AI-based controller whenever it results in a violation of the desired security property; and 3) a safety-security monitor

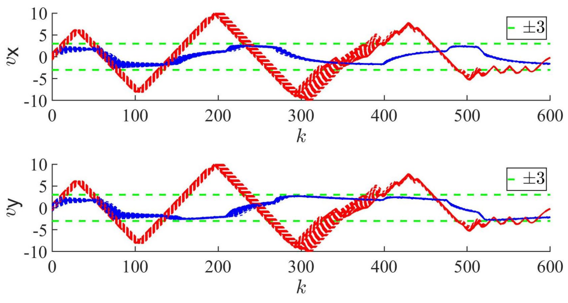

Fig. 5. Trajectories of the quadrotor's velocities with (blue) and without using the Safe-Sec-visor architecture (red).

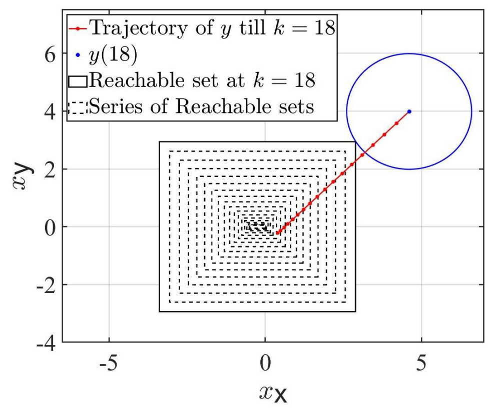

Fig. 6. A trajectory of the system without using the Safe-Sec-visor architecture. This trajectory reveals that the quadrotor started from the secret region according to Theorem 3.6. Concretely, the distance between y at time step $k = 18$ and the k-step reachable sets of the quadrotor from the set $\underline{\textbf{Proj}}(\mathcal{R}')$ is larger than $\delta = 2$ (the radius of the blue circle is 2).

that provides control input enforcing the overall safety and security of the system whenever the AI-based controller is rejected. Concretely, given a discrete-time control system together with the desired safety and security properties, we provide conditions under which one can construct (augmented) control barrier functions with respect to these properties. On top of these functions, we propose the construction of the Safe-Sec-visor architecture and the formal guarantees regarding safety and security provided by this architecture. Additionally, we also discuss how to identify the violation of the desired security property given a state-run generated by the systems. Finally, the effectiveness of the proposed methodologies is demonstrated through a case study on a quadrotor control problem. For future work, we are interested to extend this work by deploying more general safety properties (e.g., linear temporal logic properties [42], instead of invariance properties in the current work), and by exploring other notions of security properties [6] for complex CPS.

Appendix

Proof of Theorem 3.5: Firstly, according to [43, Lemma 4.1], one can verify that $\{x \in \mathbb{R}^n | x^\top Q^{-1} x \leq 1\} \subseteq X \setminus X_d$ and $\{(x,\hat{x}) \in \mathbb{R}^n \times \mathbb{R}^n | [x;\hat{x}]^\top Q_o^{-1}[x;\hat{x}] \leq 1\} \subseteq \bar{R}$ hold if and only if (III.4) and (III.6) hold, respectively, with $\bar{\mathcal{R}}$ as in (III.2). Therefore, one gets

$$\mathcal{S} := \{x \in \mathbb{R}^n | x^\top Q^{-1} x \leq c_1\} \subseteq X \setminus X_d, \quad \text{(A.1)}$$

$$\mathcal{S}_O := \{(x,\hat{x}) \in \mathbb{R}^n \times \mathbb{R}^n | [x;\hat{x}]^\top Q_o^{-1}[x;\hat{x}] \leq c_2\} \subseteq \bar{R}, \quad \text{(A.2)}$$

for all $c_1, c_2 \in (0,1]$. Having (A.1) and (A.2), Theorem 3.5 can be proved by showing

1) (**Cond.1**) $\forall x_0 \in X_0$, one has $\mathbf{x}_{x_0,\nu}(k) \in \mathcal{S}$, with $\nu(k)$ generated by running Algorithm 1, $\forall k \in \mathbb{N}$;
2) (**Cond.2**) $\forall x_0 \in X_0 \cap X_S$, $\exists \hat{x}_0 \in X_0 \setminus X_S$, with $||h(x_0) - h(\hat{x}_0)|| \leq \delta$, one has $(\mathbf{x}_{x_0,\nu}(k), \mathbf{x}_{\hat{x}_0,\hat{\nu}}(k)) \in \mathcal{S}_O$, $\forall k \in \mathbb{N}$, with $\nu(k)$ generated by Algorithm 1.

As a key insight, one can readily verify that (**Cond.1**) implies that Σ is safe since (A.1) implies $\mathcal{S} \cap X_d = \emptyset$. Meanwhile, (A.2) indicates that $\mathcal{S}_O \cap \{(x,\hat{x}) \in X \times X | \; ||h(x) - h(\hat{x})|| > \delta\} = \emptyset$ so that (**Cond.2**) implies that Σ is δ-approximate initial-state opaque.

Firstly, since $u_{\text{AI}}(k)$ will only be accepted if $\mathcal{B}(f(x(k), u_{\text{AI}}(k))) \leq 0$, we show (**Cond.1**) holds by showing (III.3) implies that there exists $c_1 \in (0,1]$ such that $\forall x_0 \in \mathcal{S}$, one has $\mathbf{x}_{x_0,\nu}(k) \in \mathcal{S}$, with $\nu(k) := \bar{K}(x(k))Q^{-1}x(k)$, $\forall k \in \mathbb{N}$, and $\mathcal{S}$ as in (A.1). Consider a controller $u(x) := K(x)x$, with $K(x) \in \mathcal{P}(x)$. For all $x \in \mathcal{S}$, one has $A\mathcal{H}(x)x + B\mathcal{U}(x)u \in \mathcal{S}$ if $P - (A\mathcal{H}(x) + B\mathcal{U}(x)K(x))^\top P(A\mathcal{H}(x) + B\mathcal{U}(x)K(x)) \succeq 0$ holds $\forall x \in \mathbb{R}^n$, with $P = Q^{-1}$, or, equivalently, $Q - (A\mathcal{H}(x)Q + B\mathcal{U}(x)\bar{K}(x))^\top P(A\mathcal{H}(x)Q + B\mathcal{U}(x)\bar{K}(x)) \succeq 0$ holds $\forall x \in \mathbb{R}^n$, with $\bar{K}(x) = K(x)Q$. Then, considering the Schur complement [44] of Q, one has this matrix inequality holds if (III.3) holds. Since $0_n \in \mathcal{S}$, and $u(x) = K(x)x = 0$ when $x = 0_n$, there must exists $c_1 \in \mathbb{R}_{>0}$ such that $u(x) \in U$, for all $x \in \mathcal{S}$. Therefore, one has (**Cond.1**) holds since $X_0 \subset \mathcal{S}$ by (**cd.2**).

Note that $u_{\text{AI}}(k)$ will only be accepted if the optimization problem OP in Algorithm 1 is feasible, and similar to the discussion above, one can readily show that (III.5) and (**cd.4**) in Definition 3.3 implies that there exist $c_2 \in (0,1]$ such that $\forall (x_0, \hat{x}_0) \in \mathcal{R}_0$, one has $(\mathbf{x}_{x_0,\nu}(k), \mathbf{x}_{\hat{x}_0,\hat{\nu}}(k)) \in \mathcal{S}_O$, with $\nu(k) := \bar{K}(x(k))Q^{-1}x(k)$, $\hat{\nu}(k) := \bar{K}_o(x(k), \hat{x}(k))Q_o^{-1}[x(k); \hat{x}(k)]$, $\forall k \in \mathbb{N}$, and $\mathcal{S}_O$ as in (A.2). To complete the proof, we still need to show that $\forall x_0 \in X_0 \cap X_S$, $\exists \hat{x}_0 \in X_0 \setminus X_S$, with $||h(x_0) - h(\hat{x}_0)|| \leq \delta$, such that $(x_0, \hat{x}_0) \in \mathcal{R}_0$. Consider any arbitrary secret initial state $x_0 \in X_0 \cap X_s$. By the assumption as in (II.7), for all $x_0 \in X_0 \cap X_s$, we get that there exists $\hat{x}_0 \in X_0 \setminus X_s$ such that $||h(x) - h(x_0)|| \leq \delta$. This indicates that the set $\mathcal{R}_0^{\text{init}}$ is not empty. Therefore, consider $\mathcal{R}_0$ satisfying (**cd.4**) in Definition 3.3, one has $\forall x_0 \in X_0 \cap X_s$, $\exists \hat{x}_0 \in X_0 \setminus X_s$, with $||h(x_0) - h(\hat{x}_0)|| \leq \delta$, such that $(x_0, \hat{x}_0) \in \mathcal{R}_O$, which completes the proof. ■

Proof of Theorem 3.6. If $\min_{\hat{x} \in P(k)} ||h(x_k) - h(\hat{x})|| > \delta$, then $\forall \hat{x} \in P(k)$, one has $||h(x_k) - h(\hat{x})|| > \delta$. This implies that for the given state run $\mathbf{x}_{x_0,\nu}$, for any $\hat{x}_0 \in X_0 \setminus X_S$ with $||h(x_0) - h(\hat{x}_0)|| \leq \delta - \epsilon$, one has $||h(\mathbf{x}_{x_0,\nu}(k)) - h(\mathbf{x}_{\hat{x}_0,\hat{\nu}}(k))|| > \delta$ for any arbitrary $\mathbf{x}_{\hat{x}_0,\hat{\nu}}$ started from $\hat{x}_0$. Therefore, the dt-CS Σ is not δ-approximate initial-state opaque according to Definition 2.4, which completes the proof. ■

References

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[2] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 269–286.

[3] K.-D. Kim and P. R. Kumar, "Cyber–physical systems: A perspective at the centennial," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1287–1308, 2012.

[4] S. Liu, A. Trivedi, X. Yin, and M. Zamani, "Secure-by-construction synthesis of cyber-physical systems," *Annual Reviews in Control*, vol. 53, pp. 30–50, 2022.

[5] H. Sandberg, S. Amin, and K. Johansson, "Cyberphysical security in networked control systems: An introduction to the issue," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 20–23, 2015.

[6] S. M. Dibaji, M. Pirani, D. B. Flamholz, A. M. Annaswamy, K. H. Johansson, and A. Chakrabortty, "A systems and control perspective of cps security," *Annual reviews in control*, vol. 47, pp. 394–411, 2019.

[7] B. Zhong, M. Zamani, and M. Caccamo, "Sandboxing controllers for stochastic cyber-physical systems," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2019, pp. 247–264.

[8] B. Zhong, A. Lavaei, H. Cao, M. Zamani, and M. Caccamo, "Safe-visor architecture for sandboxing (ai-based) unverified controllers in stochastic cyber–physical systems," *Nonlinear Analysis: Hybrid Systems*, vol. 43, p. 101110, 2021.

[9] C. Reis, A. Barth, and C. Pizano, "Browser security: lessons from google chrome," *Communications of the ACM*, vol. 52, no. 8, pp. 45–49, 2009.

[10] S. Huh and I. Yang, "Safe reinforcement learning for probabilistic reachability and safety specifications: A Lyapunov-based approach," *arXiv:2002.10126*, 2020.

[11] M. Kazemi and S. Soudjani, "Formal policy synthesis for continuous-state systems via reinforcement learning," in *International Conference on Integrated Formal Methods*. Springer, 2020, pp. 3–21.

[12] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer Science & Business Media, 2009.

[13] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 89.

[14] B. Zhong, A. Lavaei, M. Zamani, and M. Caccamo, "Automata-based controller synthesis for stochastic systems: A game framework via approximate probabilistic relations," *Automatica*, vol. 147, p. 110696, 2023.

[15] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[16] B. Zhong, M. Zamani, and M. Caccamo, "Formal synthesis of controllers for uncertain linear systems against omega-regular properties: A set-based approach," *IEEE Transactions on Automatic Control*, pp. 1–16, 2023.

[17] X. Yin, M. Zamani, and S. Liu, "On approximate opacity of cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 66, no. 4, pp. 1630–1645, 2020.

[18] S. Liu and M. Zamani, "Verification of approximate opacity via barrier certificates," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1369–1374, 2020.

[19] ——, "Compositional synthesis of opacity-preserving finite abstractions for interconnected systems," *Automatica*, vol. 131, p. 109745, 2021.

[20] B. Zhong, S. Liu, M. Zamani, and M. Caccamo, "Secure-by-construction controller synthesis via control barrier functions," *IFAC-PapersOnLine*, 2023, IFAC World Congress, to appear.

[21] B. Zhong, S. Liu, M. Caccamo, and M. Zamani, "Secure-by-construction synthesis for control systems," *arXiv:2307.02564*, 2023.

[22] A. Clavière, E. Asselin, C. Garion, and C. Pagetti, "Safety verification of neural network controlled systems," pp. 47–54, 2021.

[23] L. Humphrey, B. Könighofer, R. Könighofer, and U. Topcu, "Synthesis of admissible shields," in *Hardware and Software: Verification and Testing. HVC 2016. Lecture Notes in Computer Science*. Springer, 2016, pp. 134–151.

[24] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[25] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 176–188, 2021.

[26] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[27] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: verifying safety properties of hybrid systems with neural network controllers," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 169–178.

[28] L. Sha, "Using simplicity to control complexity," *IEEE Software*, pp. 20–28, 2001.

[29] S. Bak, T. T. Johnson, M. Caccamo, and L. Sha, "Real-time reachability for verified simplex design," in *2014 IEEE Real-Time Systems Symposium*. IEEE, 2014, pp. 138–148.

[30] B. Zhong, H. Cao, M. Zamani, and M. Caccamo, "Towards safe ai: Sandboxing dnns-based controllers in stochastic games," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 15 340–15 349.

[31] G. Chesi, "LMI techniques for optimization over polynomials in control: a survey," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2500–2510, 2010.

[32] H. Khalil, *Nonlinear Systems*, ser. Pearson Education. Prentice Hall, 2002.

[33] S. Lafortune, F. Lin, and C. Hadjicostis, "On the history of diagnosability and opacity in discrete event systems," *Annual Reviews in Control*, vol. 45, pp. 257–266, 2018.

[34] A. Saboori and C. Hadjicostis, "Verification of initial-state opacity in security applications of discrete event systems," *Information Sciences*, vol. 246, pp. 115–132, 2013.

[35] A. Ghaffari, "Analytical design and experimental verification of geofencing control for aerial applications," *IEEE/ASME Transactions on Mechatronics*, vol. 26, pp. 1106–1117, 2021.

[36] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.3.6*, 2019. [Online]. Available: http://docs.mosek.com/9.0/toolbox/index.html

[37] G. Chesi, *Domain of attraction: analysis and control via SOS programming*. Springer, 2011, vol. 20.

[38] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*. IEEE, 2004, pp. 284–289.

[39] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *Proc. of the European Control Conference*, Zürich, Switzerland, July 17–19 2013, pp. 502–510, http://control.ee.ethz.ch/~mpt.

[40] M. Althoff, "An introduction to cora 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015, p. 120–151.

[41] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations(Poster)*, 2016.

[42] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science*. IEEE, 1977, pp. 46–57.

[43] D. Seto and L. Sha, "An engineering method for safety region development," Carnegie-Mellon Univ. Pittsburgh PA Software Engineering Inst., Tech. Rep., 1999.

[44] F. Zhang, *The Schur complement and its applications*. Springer Science & Business Media, 2006, vol. 4.