Automatica 159 (2024) 111330

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Sensor deception attacks against security in supervisory control systems☆

Jingshi Yao, Shaoyuan Li, Xiang Yin *

Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China

ARTICLE INFO

Article history:
Received 16 June 2022
Received in revised form 20 June 2023
Accepted 21 August 2023
Available online xxxx

Keywords:
Discrete-event systems
Sensor attacks
Information-flow security
Opacity

ABSTRACT

This paper investigates the problem of synthesizing sensor deception attacks against security in the context of supervisory control of discrete-event systems (DES). We consider a DES plant controlled by a supervisor, whose security requirement is to maintain the *initial-secret* of the system undetected, i.e., it does not want to reveal the fact that it was initiated from a secret state. On the other hand, there exists an *active attacker* that can tamper with the observations received by the supervisor by, e.g., hacking into the communication channel between the sensors and the supervisor. The objective of the attacker is to deceive the supervisor such that the initial-secret is revealed due to incorrect control actions. We investigate the problem from the attacker's point of view and focus on synthesizing attack strategies that threaten the security of the system. We consider two levels of success of the attacker: one requires that the attacker can detect the initial-secret of the system "*almost surely*" and the other only requires that the attacker has the *possibility* to detect the initial-secret of the system. For both cases, we present algorithms for synthesizing successful attack strategies. Our approach is based on the All Attack Structure (AAS) which records state estimates for both the supervisor and the attacker. Structural properties of the security requirements are also leveraged to reduce the synthesis complexity. A running academic example is provided to illustrate the proposed synthesis procedures.

1. Introduction

With the developments of computation and communication technologies, cyber–physical systems (CPS) has become the new generation of engineering systems with computation devices embedded in physical dynamics. Particularly, in cyber–physical control systems, distributed sensors and actuators can exchange information in real-time, which on the one hand enables more flexible and intelligent control architectures, but on the other hand, makes security and privacy considerations increasingly more important in the analysis and design of CPS (Basilio, Hadjicostis, & Su, 2021; Carvalho, Wu, Kwong, & Lafortune, 2018; He, Ma, & Tang, 2021; Liu, Trivedi, Yin, & Zamani, 2022; Ma & Cai, 2022).

In this paper, we investigate security issues in CPS whose high-level behaviors are abstracted as discrete-event systems (DES). We focus on an important class of information-flow secure property called *opacity*, which captures whether or not some "secret" of the system can be inferred by an outsider via its observations. The notion of opacity has drawn much attention in the context of DES in the past few years due to its wide applications in many engineering systems such as autonomous robots (Yang, Yin, Li, & Zamani, 2020; Yu, Yin, Li, & Li, 2022), cloud computing (Zeng & Koutny, 2021) and web services (Bourouis, Klai, Ben Hadj-Alouane, & El Touati, 2017). Also, many different notions of opacity have been investigated for different security requirements such as initial-state opacity, current-state opacity, K-step opacity and infinite-step opacity. The reader can refer to Balun and Masopust (2021), Jacob, Lesage, and Faure (2016), Lin (2011), Wintenberg, Blischke, Lafortune, and Ozay (2022), Wu and Lafortune (2013) for different notions and their comparisons. In this work, we will focus on the notion of *initial-state opacity* (Lai, Lahaye, & Li, 2021; Saboori & Hadjicostis, 2013), which requires that the system can never reveal the fact when it was initiated from a set of secret states.

In general, an uncontrolled open-loop system may not naturally satisfy some desired requirements such as opacity. And *supervisors* are usually designed to restrict the behavior of the system to ensure the information-flow security of the closed-loop system under control. In the literature on DES, many algorithms have been developed for designing opacity-enforcing supervisors;

☆ This work was supported by the NSFC grants (62061136004, 62173226, 61833012). The material in this paper was partially presented at the 61st IEEE Conference on Decision and Control, Dec. 6–9, 2022, Cancún, Mexico. This paper was recommended for publication in revised form by Associate Editor Michel Reniers under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: yaojingshi@sjtu.edu.cn (J. Yao), syli@sjtu.edu.cn (S. Li), yinxiang@sjtu.edu.cn (X. Yin).

https://doi.org/10.1016/j.automatica.2023.111330
0005-1098/

see, e.g., Dubreil, Darondeau, and Marchand (2010), Saboori and Hadjicostis (2011a), Tong, Li, Seatzu, and Giua (2018), Xie, Yin, and Li (2021), Yin and Lafortune (2016). However, all existing opacity-enforcing supervisory control systems are designed for the nominal setting in the sense that no active attack exists. Many recent works show that in networked control systems, supervisors are usually subject to active and malicious attacks (Fritz & Zhang, 2018; Rashidinejad et al., 2019; Wakaiki, Tabuada, & Hespanha, 2019), which may destroy the desired closed-loop properties in the nominal setting. Particularly, the sensors of the supervisor may be compromised under *sensor deception attacks* (Alves, Pena, & Rudie, 2021; Khoumsi, 2019; Lin & Su, 2021; Lin, Tai, Zhu, & Su, 2021; Meira-Góes, Kang, Kwong, & Lafortune, 2020; Meira-Góes, Lafortune, & Marchand, 2021; Su, 2018; Wang, Li, Yu, Wu, & Li, 2021; Zhang, Li, Seatzu, & Giua, 2018) such that the supervisor may receive fictitious observation tampered with by the attacker. Then based on the incorrect information, the supervisor can be misled to take incorrect actions, which may reveal its secret.

In this paper, we investigate *from the attacker's point of view*. Specifically, we assume that there already exists a well-designed supervisor for the plant such that the closed-loop system is initial-state opaque without attacks on the communication channel, i.e., when the outsider is purely passive and eavesdropping. However, we assume that an *active attacker* can further deceive the supervisor by tampering with some observable events it receives. Our objective is to synthesize such a sensor deception attacker such that (i) it can successfully mislead the supervisor to reveal the initial-secret of the system; and (ii) at the same time, maintain itself undetected by the supervisor. Specifically, we consider two levels of success of the attacker: one is referred to as the *strong attacker*, which requires that the attacker can almost surely detect the initial-secret of the system, and the other is referred to as the *weak attacker*, which only requires that the attacker has the *possibility* to detect the initial-secret of the system. To solve the attacker synthesis problems, we build an information-flow structure called the *All Attack Structure* (ASS) that embeds all feasible attack strategies in it. Based on this structure, we show how to effectively extract attacker strategies for both the strong case and the weak case.

Our work is most related to Meira-Góes et al. (2020), Su (2018), where the authors also investigate how to synthesize sensor deception attackers against given supervisory control systems. However, compared with these two works, the present paper has the following significant differences:

- First, the attack objective considered in Meira-Góes et al. (2020), Su (2018) is the violation of safety, i.e., all *internal* strings generated by the system are legal. In this article, however, we consider the violation of security as the attack objective, which is a hyper-property defined over the *information-flow* of generated behaviors. Specifically, we consider the exposure of initial secret of the system as the violation of security. To our knowledge, the synthesis of active attackers against security requirements has not yet been studied in the context of supervisory control of DES.
- Second, due to the different attack objectives, we can further leverage *structural properties* for information-flow security to further mitigate the complexity of the synthesis procedures. Such structural properties are related to the initial-secret of the system, which do not exist in the scenarios of Meira-Góes et al. (2020), Su (2018).
- Finally, our work investigates the synthesis of strategies for both weak attackers and strong attackers. The setting in Meira-Góes et al. (2020) is similar to the case of weak attackers, which is fundamentally easier than the case of strong attackers. The setting in Su (2018) is similar to the case of strong attackers. However, a normality-like sufficient condition is imposed to guarantee the solvability of the problem. Our synthesis algorithm here is both sound and complete without any restriction by considering attackers with non-deterministic strategies.

The remaining parts of this paper are organized as follows. We first present some necessary preliminaries in Section 2. Then we formulate the attacker synthesis problems in Section 3. In Section 4, an information structure called the All Attack Structure is proposed, which is further simplified in Section 5 by leveraging structural properties in security. Then we present synthesis algorithms in Section 6. Finally, we conclude the paper in Section 7. The preliminary version of this paper is presented in Yao, Yin, and Li (2022) without proofs. Here, we provide complete proofs and detailed examples. Furthermore, Yao et al. (2022) only consider weak attackers with deterministic strategies. Here, we consider a more general setting, where attackers are allowed to use non-deterministic strategies, and both weak and strong attackers are synthesized.

2. Preliminary

In this section, we briefly introduce some basic knowledges in supervisory control, state estimation and information-flow security.

2.1. Supervisory control of discrete event systems

Let Σ be a finite set of symbols. A string over Σ is a finite sequence $s = \sigma_1 \cdots \sigma_n, \sigma_i \in \Sigma$. We denote by Σ^* the set of all finite strings over Σ including the empty string ϵ. For each string $s = \sigma_1 \cdots \sigma_n \in \Sigma^*$, we denote by $|s| = n$ the length of s and $s_{[i]} = \sigma_1 \cdots \sigma_i$ denotes the prefix of s with length $i \leq n$. We define $\Sigma^\epsilon = \Sigma \cup \{\epsilon\}$.

A DES is modeled by a finite-state automaton

$$G = (X, \Sigma, \delta, X_0),$$

where X is a finite set of states; Σ is the set of events; $\delta : X \times \Sigma \to X$ is the (partial) transition function and $X_0 \subseteq X$ is the set of all possible initial states. The transition function can also be extended to $\delta : X \times \Sigma^* \to X$. We define $\Delta_G(x) = \{\sigma \in \Sigma : \delta(x, \sigma)!\}$ as the set of events feasible at state $x \in X$ in G, where ! means "is defined". The language generated from $x_0 \in X_0$ by G is defined as $\mathcal{L}(G, x_0) = \{s \in \Sigma^* : \delta(x_0, s)!\}$; we define the language generated by G as $\mathcal{L}(G) = \cup_{x_0 \in X_0} \mathcal{L}(G, x_0)$.

In the supervisory control theory, Σ is partitioned by:

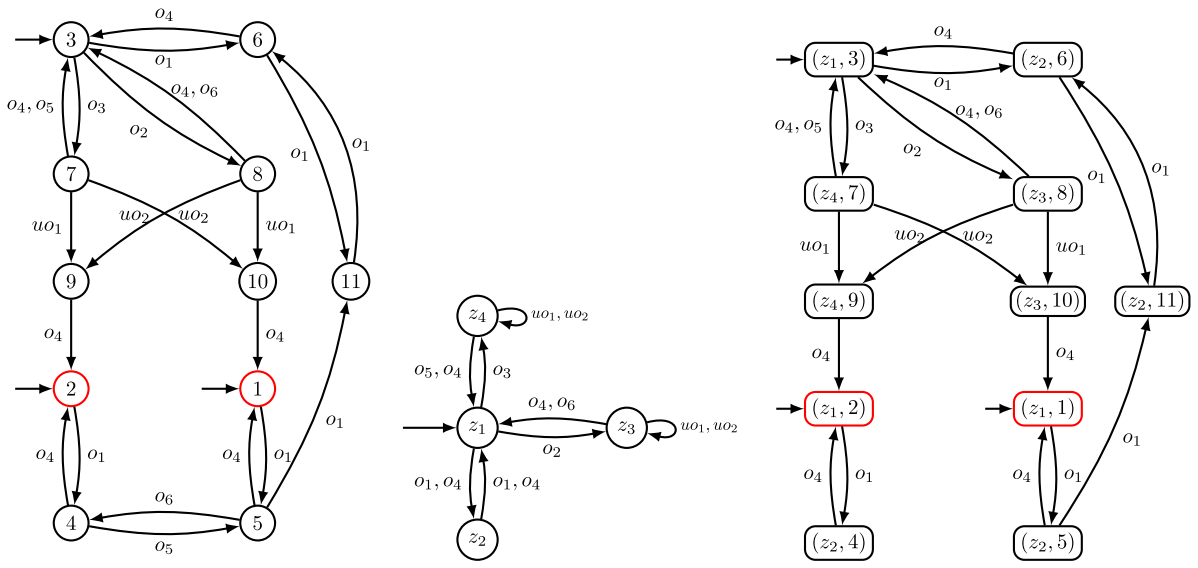
$$\Sigma = \Sigma_o \dot\cup \Sigma_{uo} = \Sigma_c \dot\cup \Sigma_{uc},$$

where Σ_o (respectively, Σ_c) is the set of observable (respectively, controllable) events, and Σ_{uo} (respectively, Σ_{uc}) is the set of unobservable (respectively, uncontrollable) events. Note that there is no relationship between controllable events and observable events in general. The natural projection is $P : \Sigma^* \to \Sigma_o^*$ which replaces each unobservable event in a string by the empty string ϵ. Natural projection can also be extended to $P : 2^{\Sigma^*} \to 2^{\Sigma_o^*}$ in the usual manner.

A partial-observation supervisor is a function

$$S : P(\mathcal{L}(G)) \to \Gamma,$$

where $\Gamma = \{\gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma\}$ is the set of feasible control patterns. We use notation S/G to represent the closed-loop system under control. The generated language of S/G starting from $x_0 \in X_0$, denoted by $\mathcal{L}(S/G, x_0)$, is defined recursively by:



(a) System G (b) Supervisor H (c) Closed-loop system $G \times H$

Fig. 1. System G and supervisor H, where $\Sigma_o = \{o_1, o_2, o_3, o_4, o_5, o_6\}$, $\Sigma_c = \Sigma \setminus \{o_4\}$, $X_0 = \{1, 2, 3\}$, and $X_{sec} = \{1, 2\}$. States marked in red are secret states. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- $\epsilon \in \mathcal{L}(S/G, x_0)$; and
- for any $s \in \Sigma^*, \sigma \in \Sigma$, we have $s\sigma \in \mathcal{L}(S/G, x_0)$ iff $[s \in \mathcal{L}(S/G, x_0)] \wedge [s\sigma \in \mathcal{L}(G, x_0)] \wedge [\sigma \in S(P(s))]$.

We assume without loss of generality that the supervisor does not know exactly the initial state of the system. Then the language generated by the controlled system S/G is defined by $\mathcal{L}(S/G) = \cup_{x_0 \in X_0} \mathcal{L}(S/G, x_0)$.

Throughout the paper, we assume that the supervisor S can be *recognized* by a deterministic finite-state automaton $H = (Z, \Sigma, \xi, z_0)$ such that

(i) $(\forall z \in Z)(\forall \sigma \in \Sigma)[\delta(z, \sigma) \neq z \Rightarrow \sigma \in \Sigma_o]$; and
(ii) $(\forall s \in \mathcal{L}(S/G))[\Delta_H(\xi(z_0, s)) = S(P(s))]$.

Then the closed-loop language can also be computed by $\mathcal{L}(S/G) = \mathcal{L}(G \times H)$, where $\times$ is the standard product composition of automata; see, e.g., Cassandras and Lafortune (2021).

Example 1. Consider system G shown in Fig. 1(a), where $\Sigma_o = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ is the set of observable events, $\Sigma_c = \Sigma \setminus \{o_4\}$ is the set of controllable events and $X_0 = \{1, 2, 3\}$ is the set of initial states. Let us consider a supervisor S which is realized by automaton H shown in Fig. 1(b). Then the closed-loop system under control S/G is realized by automaton $G \times H$ shown in Fig. 1(c).

Given a set of states $q \subseteq X$ representing the set of possible states, a control pattern $\gamma \subseteq \Sigma$ representing the set of enabled events and an observable event $\sigma \in \Sigma_o$ representing the new observation. We denote the *unobservable reach* of q under γ by

$$\mathsf{UR}_\gamma(q) := \{\delta(x, s) \in X : x \in q, s \in (\Sigma_{uo} \cap \gamma)^*\},$$

and the *observable reach* of q upon σ by

$$\mathsf{NX}_\sigma(q) := \{\delta(x, \sigma) \in X : x \in q\}.$$

We also define $\mathsf{NX}_\epsilon(q) = q$ for technical reason. Then the set of *next observable events* from q under control pattern γ is denoted as:

$$\mathcal{O}(q, \gamma) = \left\{ \sigma \in \Sigma_o \cap \gamma : \begin{array}{c} \exists x \in q, w \in (\Sigma_{uo} \cap \gamma)^* \\ \text{s.t. } \delta(x, w\sigma)! \end{array} \right\}$$

2.2. Initial secret of the control system

Given system G and supervisor S, upon the occurrence of observable string $\alpha \in P(\mathcal{L}(S/G))$, the supervisor can estimate the current-state or the initial-state of the system. We denote by $\mathcal{E}_S^C(\alpha)$ and $\mathcal{E}_S^I(\alpha)$, respectively, the current-state estimate and the initial-state estimate of the supervisor upon observation α, i.e.,

$$\mathcal{E}_S^C(\alpha) = \left\{ x \in X : \begin{array}{c} \exists x_0 \in X_0, s \in \mathcal{L}(S/G, x_0) \\ \text{s.t. } P(s) = \alpha \wedge \delta(x_0, s) = x \end{array} \right\}$$

$$\mathcal{E}_S^I(\alpha) = \{x_0 \in X_0 : \exists s \in \mathcal{L}(S/G, x_0) \text{ s.t. } P(s) = \alpha\}$$

Particularly, the current-state estimate $\mathcal{E}_S^C(\alpha)$ can be computed recursively by:

- $\mathcal{E}_S^C(\epsilon) = \mathsf{UR}_{S(\epsilon)}(X_0)$; and
- for any $\alpha\sigma \in P(\mathcal{L}(S/G))$, where $\sigma \in \Sigma_o$, we have $\mathcal{E}_S^C(\alpha\sigma) = \mathsf{UR}_{S(\alpha\sigma)}(\mathsf{NX}_\sigma(\mathcal{E}_S^C(\alpha)))$.

Furthermore, using the notion of current-state estimate, we know recursively that, for any $\alpha \in \Sigma_o^*, \sigma \in \Sigma_o$, we have $\alpha\sigma \in P(\mathcal{L}(S/G))$ iff (i) $\alpha \in P(\mathcal{L}(S/G))$; and (ii) $\sigma \in S(\alpha)$; and (iii) $\mathsf{NX}_\sigma(\mathcal{E}_S^C(\alpha)) \neq \emptyset$.

In some cases, for example, due to insecure communications, the observable output of the system may also be available to an outsider. Now, we consider a passive intruder with the following capabilities:

A1 It knows both the system model G and the functionality of supervisor S;
A2 It can also observe the occurrences of events in Σ_o.

Essentially, the passive intruder has exactly the same knowledge about the system as the supervisor.

Furthermore, we assume that the system has some "secret" that should not be revealed to the outside world. In this work, the "secret" is defined as a special subset of initial states, denoted by $X_{sec} \subset X_0$, called the *secret initial states*. Then upon the occurrence of sequence $\alpha \in P(\mathcal{L}(S/G))$, we say the *initial secret* (IS) of system G is revealed (or, *detected* by the intruder) if $\mathcal{E}_S^I(\alpha) \subseteq X_{sec}$, i.e., the outside world will determine for sure that it is initiated from a secret state. In practice, the open-loop system may reveal its IS from some observations. Therefore, supervisor S is usually designed to

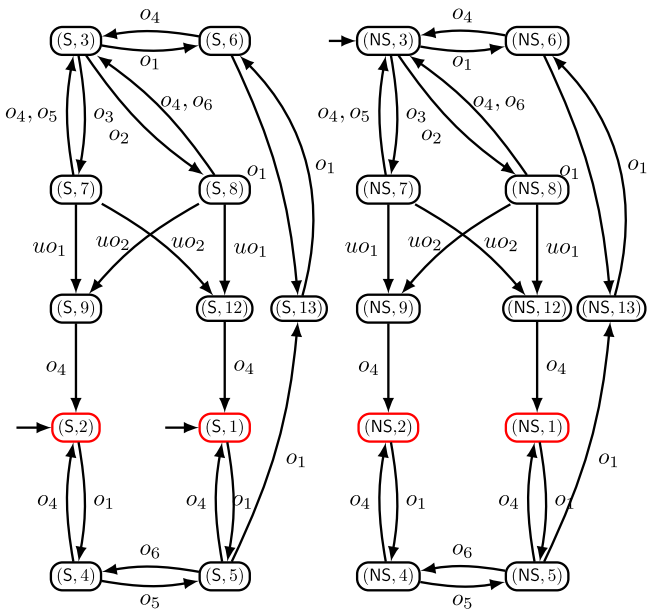


Fig. 2. Augmented system $\tilde{G}$ for system G in Fig. 1(a).

restrict the behavior of the system to guarantee that the IS of the system cannot be detected by the intruder from any observation; see, e.g., Dubreil et al. (2010), Saboori and Hadjicostis (2011a), Tong et al. (2018), Xie et al. (2021), Yin and Lafortune (2016) for how to synthesize such a supervisor.

The *augmented system* of G is defined as an automaton

$$\tilde{G} = (\tilde{X}, \Sigma, \tilde{\delta}, \tilde{X}_0),$$

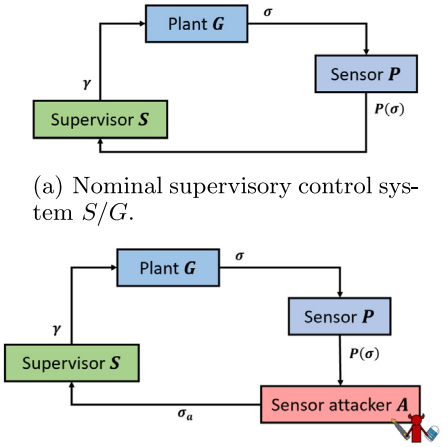
where $\tilde{X} \subseteq \{\mathsf{S}, \mathsf{NS}\} \times X$ is the set of states, $\tilde{X}_0 = \{\mathsf{S}\} \times X_{sec} \cup \{\mathsf{NS}\} \times (X_0 \setminus X_{sec})$ is the set of initial-states, and the transition function $\tilde{\delta} : \tilde{X} \times \Sigma \to \tilde{X}$ is defined by: for any $(l, x) \in \tilde{X}$, $\sigma \in \Sigma$, we have $\tilde{\delta}((l, x), \sigma) = (l, \delta(x, \sigma))$.

Intuitively, $\tilde{G}$ simply augments the state-space of G with a label in $\{\mathsf{S}, \mathsf{NS}\}$ representing whether or not it is initiated from a secret state. Clearly, we have $\mathcal{L}(\tilde{G}) = \mathcal{L}(G)$ and $\mathcal{L}(S/\tilde{G}) = \mathcal{L}(S/G)$. It also makes sense to define the current-state estimate and initial-state estimate of the supervisor over the state-space of $\tilde{G}$ and denote them by $\tilde{\mathcal{E}}_S^C(\alpha)$ and $\tilde{\mathcal{E}}_S^I(\alpha)$ respectively. Furthermore, for any set of augmented states $\tilde{q} \subseteq \tilde{X}$, we denote by $I(\tilde{q}) = \{l : (l, x) \in \tilde{q}\} \subseteq \{\mathsf{S}, \mathsf{NS}\}$ the set of first components of $\tilde{q}$. Then for any observation $\alpha \in P(\mathcal{L}(S/G))$, we have that the IS is revealed if and only if the first component set of current-state estimate over $\tilde{G}$ is $\{\mathsf{S}\}$, i.e.,

$$\mathcal{E}_S^I(\alpha) \subseteq X_{sec} \Leftrightarrow I(\tilde{\mathcal{E}}_S^C(\alpha)) = \{\mathsf{S}\}. \tag{1}$$

For the sake of clarity, we use notations $\widetilde{\mathsf{NX}}_\sigma(\tilde{q})$ and $\widetilde{\mathsf{UR}}_\gamma(\tilde{q})$ and $\tilde{\mathcal{O}}(\tilde{q}, \gamma)$ to denote the observable reach, the unobservable reach and the next observable events over the augmented state-space, respectively.

Example 2. For system G in Fig. 1(a), its augmented system $\tilde{G}$ is shown in Fig. 2. Note that $\tilde{G}$ has two disconnected parts, where the left (respectively, right) part corresponds to the case where the system starts from secret (respectively, non-secret) initial states. Therefore, these two parts are the same except their initial states. Without the supervisor, if the system is initiated from state 1 and the sequence o_1o_6 is observed, we have $\tilde{\mathcal{E}}_S^C(o_1o_6) = \{(\mathsf{S}, 4)\}$ and $I((\mathsf{S}, 4)) = \{\mathsf{S}\}$, i.e., the intruder knows for sure that the system was from a secret-state. However, under the supervision of S shown in Fig. 1(b), the closed-loop system under control is shown as Fig. 1(c) where string o_1o_6 is no longer feasible in $\mathcal{L}(S/\tilde{G}) = \mathcal{L}(S/G)$ and the IS will not be detected.



(a) Nominal supervisory control system S/G.

(b) Supervisory control system S_A/G under attack.

Fig. 3. Illustration of sensor attacks.

3. Active attack against initial secret

In this section, we formulate the attacker synthesis problems that we solve in this paper.

3.1. Motivating example

In the standard problem concerning initial secret, the intruder is assumed to be *passive* in the sense that it can only "eavesdrop" on the observations without interfering with the observations received by the supervisor. In Example 2, we have provided an example, where the supervisor successfully protects the initial secret of the closed-loop system from being revealed to the passive intruder, that is, upon any observation of the closed-loop system S/G, the initial-state estimate of the intruder will not be a subset of secret initial states.

However, as depicted in Fig. 3(b), in networked control systems, some powerful attackers may further have the capability to *actively* tamper with the observations sent from the sensors to the supervisor in order to mislead the feedback decisions. Such an attacker model is referred to as the *sensor deception attacks* (Meira-Góes et al., 2020; Su, 2018). The following example shows that, in the case of sensor deception attacks, the attacker may actively mislead a supervisor, which is originally IS-enforcing, to reveal its IS.

Example 3. Still, we consider system G shown in Fig. 1(a) and supervisor S recognized by automaton shown in Fig. 1(b). We assume that there exists an attacker A as depicted in Fig. 3(b). The strategy of attacker A is simply to *replace* the supervisor's observation of event o_1 by event o_2. Recall that, for supervisor S in Example 1, we have $S(o_2) = \{o_4, o_6, uo_1, uo_2\}$. Therefore, under attacker A, when the system starts from initial state $1 \in X_{sec}$ and moves to state 5 via event o_1, the observation of the supervisor is o_2 rather than o_1 due to the presence of the attacker A. Since event o_6 is enabled in $S(o_2)$, sequence o_1o_6 can still occur and upon which the attacker knows for sure that the system was initiated from secret state 1, i.e., the IS of the system is revealed.

In this work, we will focus on how to synthesize such attacker A for the purpose of detecting the initial secret of a given supervisory control system.

3.2. Strategies of active sensor attacks

To formalize the above discussion, in addition to A1 and A2, we make the following assumption regarding the capability of the *active* sensor attacker.

A3 $\Sigma_v \subseteq \Sigma_o$ is the set of *vulnerable events* whose occurrences can be tampered with (either be replaced by a new event in Σ_v or be deleted) by the active sensor attacker.

In order to distinguish between the observations of the supervisor and the intruder, we denote

$$\hat{\Sigma}_o = \{\hat{\sigma} : \sigma \in \Sigma_o\} \cup \{\hat{\epsilon}\}$$

as the set of events processed by the attacker, i.e., the set of events received by the supervisor. Note that, from the supervisor's point of view, it cannot distinguish between event σ and $\hat{\sigma}$. For each observable event $\sigma \in \Sigma_o$, we define the action-space of the active attacker by a mapping $\mathcal{V} : \Sigma_o \to \hat{\Sigma}_o$ such that

$$\mathcal{V}(\sigma) = \begin{cases} \{\hat{\sigma}_a : \sigma_a \in \Sigma_v\} \cup \{\hat{\epsilon}\} & \text{if } \sigma \in \Sigma_v \\ \{\hat{\sigma}\} & \text{if } \sigma \notin \Sigma_v \end{cases}.$$

That is, for each vulnerable event $\sigma \in \Sigma_v$, the attacker can either replace it by a new event $\hat{\sigma}_a$ (the supervisor observes $\sigma_a \in \Sigma_v$) or delete it (the supervisor observes nothing). However, for events that are not vulnerable, the attacker cannot change the observation.

Hence, the entire operation process of the supervisor control system under attack is as follows. Initially, the plant generates the first observable event $\sigma_1 \in \Sigma_o$ that is feasible under the control decision $S(\epsilon) \in \Gamma$. Then the attacker replaces it by a new event $\hat{\sigma}_{a1} \in \mathcal{V}(\sigma_1)$. Therefore, the supervisor observes $\sigma_{a1} \in \Sigma_o$ and issues control decision $S(\sigma_{a1}) \in \Gamma$. Then the plant generates the second observable event $\sigma_2 \in \Sigma_o \cap \Gamma$ and the attacker replaces it by some $\hat{\sigma}_{a2} \in \mathcal{V}(\sigma_2)$, and so forth.

To formalize the above process, we introduce the notion of *extended string*. Formally, an extended string is an alternating sequence of events of the following form

$$\pi = \sigma_1\hat{\sigma}_{a1}\sigma_2\hat{\sigma}_{a2}\cdots\sigma_n\hat{\sigma}_{an} \in (\Sigma\hat{\Sigma}_o)^*, \tag{2}$$

such that for any $i = 1, 2, \ldots$, if $\sigma_i \in \Sigma_o$, we have $\hat{\sigma}_{ai} \in \mathcal{V}(\sigma_i)$ and $\hat{\sigma}_{ai} = \hat{\epsilon}$ otherwise.

Then upon the occurrence of extended string $\pi = \sigma_1\hat{\sigma}_{a1}\cdots\sigma_n\hat{\sigma}_{an} \in (\Sigma\hat{\Sigma}_o)^*$, the actual internal string generated by the system is denoted by

$$\pi_{\mathsf{G}} = \sigma_1\sigma_2\cdots\sigma_n \in \Sigma^*.$$

Furthermore, the information in π available to the attacker and the supervisor are also different. Specifically, the attacker knows both each observable event and its replacements. Formally, let $1 \le i_1 < \cdots < i_m \le n$ be the indices such that $\sigma_{i_k} \in \Sigma_o$. Then the information of the attacker along extended string π is

$$\pi_{\mathsf{A}} = \sigma_{i_1}\hat{\sigma}_{ai_1}\sigma_{i_2}\hat{\sigma}_{ai_2}\cdots\sigma_{i_m}\hat{\sigma}_{ai_m} \in (\Sigma_o\hat{\Sigma}_o)^*.$$

It can be extended to a set of extended strings L as $L_{\mathsf{A}} = \{\pi_{\mathsf{A}} \in (\Sigma_o\hat{\Sigma}_o)^* : \pi \in L\}$. On the other hand, the information of the supervisor along extended string π is

$$\pi_{\mathsf{S}} = \sigma_{ai_1}\sigma_{ai_2}\cdots\sigma_{ai_m} \in \Sigma_o^*.$$

Now, it remains to specify how the attacker tamper with each actual observable event $\sigma_i \in \Sigma_o$. Formally, the strategy of an active sensor attacker is a function

$$A : (\Sigma_o\hat{\Sigma}_o)^*\Sigma_o \to 2^{\hat{\Sigma}_o},$$

such that, for any $\pi_{\mathsf{A}} \in (\Sigma_o\hat{\Sigma}_o)^*$ and $\sigma \in \Sigma_o$, we have

$$\emptyset \subset A(\pi_{\mathsf{A}}\sigma) \subseteq \mathcal{V}(\sigma).$$

Intuitively, when the attacker knows what has happened in the past, i.e., π_{A}, and observes a new observable event σ, it will decide to replace σ by *one of* the events in $A(\pi_{\mathsf{A}}\sigma)$ *non-deterministically*. Therefore, $A(\pi_{\mathsf{A}}\sigma)$ is also referred to as the decision set of the attacker for $\pi_{\mathsf{A}}\sigma$. For technical purpose, we also define $A(\epsilon) = \{\hat{\epsilon}\}$. If $A(\pi_{\mathsf{A}}\sigma)$ is always a singleton for any $\pi_{\mathsf{A}}\sigma$, then we say the attacker strategy A is deterministic; otherwise, it is non-deterministic. As we will see later, the attacker is more powerful by using non-deterministic strategies.

As depicted in Fig. 3(b), we denote by S_A/G the closed-loop system under attack. Since both the actual internal events and the tampered events are involved, we need to define the behavior of S_A/G in terms of extended strings. Then according to the process illustrated above, starting from initial state $x_0 \in X_0$, the extended language generated by S_A/G, denoted by $\mathcal{L}_e(S_A/G, x_0)$, is defined recursively as:

- $\epsilon \in \mathcal{L}_e(S_A/G, x_0)$; and
- for any extended string $\pi \in (\Sigma\hat{\Sigma}_o)^*$, a new event $\sigma \in \Sigma$, and tampered event $\hat{\sigma}_a \in \hat{\Sigma}_o$, we have $\pi\sigma\hat{\sigma}_a \in \mathcal{L}_e(S_A/G, x_0)$ iff

  (i) $\pi \in \mathcal{L}_e(S_A/G, x_0)$; and
  (ii) $\pi_{\mathsf{G}}\sigma \in \mathcal{L}(G, x_0)$; and
  (iii) $\sigma \in S(\pi_{\mathsf{S}})$; and
  (iv) $\hat{\sigma}_a \in \begin{cases} A(\pi_{\mathsf{A}}\sigma) & \text{if } \sigma \in \Sigma_o \\ \{\hat{\epsilon}\} & \text{if } \sigma \in \Sigma_{uo} \end{cases}$.

The intuition of the above definition is as follows. Condition (i) says that the prefix π is an extended string generated by the attacked system. Condition (ii) says that event σ is feasible after internal string π_{G} in G. Condition (iii) says that event σ is also enabled by the supervisor S based on what it observes, i.e., π_{S}. Finally, condition (iv) says that the observable event is tampered with following the strategy of attacker A. Then the extended language generated by S_A/G is defined as $\mathcal{L}_e(S_A/G) = \cup_{x_0\in X_0}\mathcal{L}_e(S_A/G, x_0)$, and the internal language generated by S_A/G is defined as $\mathcal{L}(S_A/G) = \{\pi_{\mathsf{G}} \in \mathcal{L}(G) : \pi \in \mathcal{L}_e(S_A/G)\}$.

3.3. State estimations under attacks

Now we discuss the state-estimate issues under active sensor attacks. Upon the occurrence of an extended string $\pi \in \mathcal{L}_e(S_A/G)$, from the supervisor's point of view, it observes $\pi_{\mathsf{S}} \in \Sigma_o^*$. If we assume that the supervisor is unaware of the presence of the attacker, then it will estimate the state based on the original closed-loop system S/G and observation π_{S}, i.e., its current and initial-state estimates are $\mathcal{E}_S^C(\pi_{\mathsf{S}})$ and $\mathcal{E}_S^I(\pi_{\mathsf{S}})$, respectively.

However, from the attacker's point of view, it has strictly more information than the supervisor. Specifically, it knows the actual observation of the system $P(\pi_{\mathsf{G}})$ and the tampered observation π_{S} sent to the supervisor. Therefore, its information is $\pi_{\mathsf{A}} \in (\Sigma_o\hat{\Sigma}_o)^*$ and it will estimate the state of the system based on the attacked closed-loop system S_A/G. Then the current-state and initial-state estimates from the attacker's point of view by observing π_{A} are, respectively

$$\mathcal{E}_A^C(\pi_{\mathsf{A}}) = \left\{x \in X : \begin{array}{c} \exists x_0 \in X_0, \exists \pi' \in \mathcal{L}_e(S_A/G, x_0) \\ \text{s.t. } \pi'_{\mathsf{A}} = \pi_{\mathsf{A}} \wedge \delta(x_0, \pi'_{\mathsf{G}}) = x \end{array}\right\}$$

$$\mathcal{E}_A^I(\pi_{\mathsf{A}}) = \left\{x_0 \in X_0 : \exists \pi' \in \mathcal{L}_e(S_A/G, x_0) \text{ s.t. } \pi'_{\mathsf{A}} = \pi_{\mathsf{A}}\right\}$$

Similar to the case of $\mathcal{E}_S^C(\alpha)$, the current-state estimate $\mathcal{E}_A^C(\pi)$ can also be computed recursively by:

- $\mathcal{E}_A^C(\epsilon) = \mathsf{UR}_{S(\epsilon)}(X_0)$; and
- for any $\pi_{\mathsf{A}} \in (\Sigma_o\hat{\Sigma}_o)^*$ such that $\mathcal{E}_A^C(\pi_{\mathsf{A}})$ has been computed, for $\pi'_{\mathsf{A}} = \pi_{\mathsf{A}}\sigma\hat{\sigma}_a \in (\Sigma_o\hat{\Sigma}_o)^*$, we have $\mathcal{E}_A^C(\pi'_{\mathsf{A}}) = \mathsf{UR}_{S(\pi_{\mathsf{S}}\sigma_a)}(\mathsf{NX}_\sigma(\mathcal{E}_A^C(\pi_{\mathsf{A}})))$.

And the current-state estimate and initial-state estimate of the attacker over the state-space of $\tilde{G}$ are denoted by $\tilde{\mathcal{E}}_A^C(\alpha)$ and $\tilde{\mathcal{E}}_A^I(\alpha)$ respectively.

Note that, by tampering with the observation arbitrarily, the attacker may expose itself. Formally, for any extended string $\pi \in \mathcal{L}_e(S_A/G)$, the supervisor will detect the presence of the attacker if it observes strings that are not in $P(\mathcal{L}(S/G))$. Then the supervisor may upgrade its security level or take further actions to protect the system from being damaged. In this work, we are only interested in the process when the attacker remains undetected and once it exposes itself, the entire process is assumed to be terminated.

Definition 1 (*Stealthy & IS-Detected Strings*)**.** Let $\pi \in (\Sigma\hat{\Sigma}_o)^*$ be an extended string such that $\pi_S = \pi'\sigma$, where $\sigma \in \Sigma_o$. We say extended string π is

- *stealthy* if $\pi_S \in P(\mathcal{L}(S/G))$; and
- *semi-stealthy* if $\pi' \in P(\mathcal{L}(S/G))$; and
- *IS-detected* if it is semi-stealthy and $\mathcal{E}_A^I(\pi_A) \subseteq X_{sec}$.

Intuitively, a semi-stealthy string is either a stealthy string or a string upon which the attack exposes itself *for the first time*. Since we assume that, once the supervisor detects the presence of the attacker, the process is terminated, hereafter we will restrict extended language $\mathcal{L}_e(S_A/G)$ only to semi-stealthy strings. Also, we note that, if $\pi \in \mathcal{L}_e(S_A/G, x_0)$ is from non-secret initial state $x_0 \notin X_{sec}$, then π cannot be IS-detected since the IS is not true.

Depending on whether the active sensor attacker can *possibly* detect the IS of the system or can *almost surely* detect the initial secret, we define two levels of success for the attacker as follows.

Definition 2 (*IS-Detectable Attackers*)**.** Given system G, supervisor S and a set of secret initial states $X_{sec} \subset X_0$, we say an attacker A is

- *weakly* IS-detectable if it can detect the IS of the system along *some possible* extended strings, i.e., there exists $\pi \in \mathcal{L}_e(S_A/G)$ that is IS-detected.
- *strongly* IS-detectable if it *always has the potential* to detect the IS of the system whenever the system is actually initiated from a secret initial state, i.e.,

$$(\forall x_0 \in X_{sec})(\forall \pi \in \mathcal{L}_e(S_A/G, x_0) : \pi \text{ is semi-stealthy})$$
$$(\exists \pi\pi' \in \mathcal{L}_e(S_A/G, x_0))[\pi\pi' \text{ is IS-detected}]$$

We say the supervisory control system S/G is weakly (respectively, strongly) IS-attackable if there exists a weakly (respectively, strongly) IS-detectable attacker A.

Intuitively, the strong IS-detectability condition is an *almost sure* reachability condition in the sense that the "possibility" that the attacked system reaches an IS-revealing information configuration will increase and converge to one as the system evolves. To be more specific, since S_A is the supervisor under attack, upon the occurrence of each extended string, the system will execute an event in the enabled set provided by the attacked supervisor. If we assume that each event in the enabled set has a non-zero probability to be chosen (which is reasonable, as otherwise it means some events are disabled), then strong IS-detectability condition can be interpreted as the ability that the attacker can detect the initial secret "with probability one" as the system evolves indefinitely from a secret state.

As we discussed above, the objective of this article is to synthesize an active attacker such that it is weakly (or strongly) IS-detectable. The attacker synthesis problem is formulated as follows.

Problem 1 (*Attacker Synthesis Problem*)**.** Given system G, observable event set $\Sigma_o \subseteq \Sigma$, supervisor S, secret initial-state set $X_{sec} \subset X_0$ and vulnerable event set $\Sigma_v \subseteq \Sigma_o$, determine if the system is weakly (respectively, strongly) IS-attackable. If so, synthesize a weakly (respectively, strongly) IS-detectable attacker $A : (\Sigma_o\hat{\Sigma}_o)^*\Sigma_o \to 2^{\hat{\Sigma}_o}$.

In order to solve the above problem, in Section 4, we will first propose an information structure that serves as the solution basis. To mitigate the complexity, this structure will be further simplified in Section 5. Then in Section 6, we will establish solution algorithms for both strong and weak cases based on the simplified structures.

4. The general all attack structures

In this section, we define the All Attack Structure (AAS) that embeds all possible attacker's strategies in it with a suitably chosen state-space, which, therefore, can serve as the basis for the solutions of our synthesis problems.

Definition 3 (*All Attack Structure*)**.** Given system $G = (X, \Sigma, \delta, X_0)$, observable events $\Sigma_o \subseteq \Sigma$, supervisor S realized by automaton $H = (Z, \Sigma, \xi, z_0)$ and vulnerable events $\Sigma_v \subseteq \Sigma_o$, the All Attack Structure M is a new deterministic finite-state automaton

$$M = (Y, \Sigma_M, f, q_0), \tag{3}$$

where

- $Y = Y_e \dot{\cup} Y_a$ is the set of states with two types:
  - $Y_e \subseteq 2^X \times 2^{\tilde{X}} \times (Z \cup \{z_{\text{att}}\})$ is the set of *environment states* and z_{att} is a new symbol related to the exposure of the attacker; and
  - $Y_a \subseteq 2^X \times 2^{\tilde{X}} \times Z \times \Sigma_o$ is the set of *attack states*;
- $y_0 = (X_0, \tilde{X}_0, z_0) \in Y_e$ is the initial state;
- $\Sigma_M = \Sigma_o \cup \hat{\Sigma}_o$ is the set of events;
- $f : Y \times \Sigma_M \to Y$ is the transition function and consists of the following two types of transitions:
  - $f_{ea} : Y_e \times \Sigma_o \to Y_a$ is the transition function from environment states to attack states defined by: for any $y = (q, \tilde{q}, z) \in Y_e$, we have
  $$\Delta_M(y) = \begin{cases} \tilde{\mathcal{O}}(\tilde{q}, \Delta_H(z)) & \text{if } z \in Z \\ \emptyset & \text{if } z = z_{\text{att}} \end{cases},$$
  and for $\sigma \in \Delta_M(y)$, we have $f_{ea}(y, \sigma) = (q, \tilde{q}, z, \sigma)$;
  - $f_{ae} : Y_a \times \hat{\Sigma}_o \to Y_e$ is the transition function from attack states to environment states defined by: for any $y = (q, \tilde{q}, z, \sigma) \in Y_a$, we have $\Delta_M(y) = \mathcal{V}(\sigma)$, and for each $\hat{\sigma}_a \in \Delta_M(y)$, we have $f_{ae}(y, \hat{\sigma}_a) = (q', \tilde{q}', z')$, where
  $$\begin{cases} q' = \mathsf{NX}_{\sigma_a}(\mathsf{UR}_{\Delta_H(z)}(q)) \\ \tilde{q}' = \widetilde{\mathsf{NX}}_{\sigma}(\widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})) \end{cases}.$$
  and
  $$z' = \begin{cases} \xi(z, \sigma_a) & \text{if } [q' \neq \emptyset] \wedge [\xi(z, \sigma_a)!] \\ z_{\text{att}} & \text{otherwise} \end{cases}.$$

The intuition of the AAS is explained as follows. The AAS essentially serves as an arena for the game between the system and the attacker, where the *system-player* plays at each *environment state* by generating a feasible observation. Specifically, each component in an environment state $(q, \tilde{q}, z) \in Y_e$ is related to, respectively, the current-state estimate of the supervisor, the (augmented) current-state estimate of the attacker and the state

of the supervisor (proven in Proposition 1). Therefore, $\Delta_H(z)$ is the control decision applied currently, and whenever a feasible observation $\sigma \in \tilde{\mathcal{O}}(\tilde{q}, \Delta_H(z))$ is generated, the game moves to an attack state simply by "remembering" σ. Note that we use state z_{att} to denote that the supervisor has detected the presence of the attacker (proven in Proposition 2), and therefore, no active event is defined at z_{att}.

The *attacker-player* plays at each *attack states* by choosing *a set of* possible tampered observations, where each time one of them is selected randomly online to play. Specifically, for an attack state $(q, \tilde{q}, z, \sigma) \in Y_a$, the first three components are the same as those in its predecessor environment state, and the last component $\sigma \in \Sigma_o$ represents the latest observation of the attacker. This actual observation σ can be replaced by any event $\hat{\sigma}_a \in \mathcal{V}(\sigma)$ and the supervisor observes σ_a. If σ_a is not defined at its supervisor state z, or not feasible for observations in S/G, the supervisor will realize the presence of the attacker immediately and hence, the third component of the state moves to z_{att}; otherwise, it will update the supervisor state based on what it observes, i.e., $z' = \xi(z, \sigma_a)$. For technical reason, we define $\mathsf{UR}_{\Delta_H(z_{\text{att}})}(q) = \emptyset$.

We note that, any string in M ending up with an environment state is an observable extended string, and we denote by $\mathcal{L}_e(M) = \{\pi \in \Sigma_M^* : f(\pi) \in Y_e\}$ the set of (observable) extended strings generated by M. Now we prove that, for any extended string $\pi \in \mathcal{L}_e(M)$, the first and the second components of environment state reached are, respectively, related to the state estimate of the supervisor and the (augmented) state estimate of the attacker.

Proposition 1. *For any extended string $\pi \in \mathcal{L}_e(M)$, let $(q, \tilde{q}, z) = f(y_0, \pi)$, we have*

$$\mathcal{E}_S^C(\pi_{\mathsf{S}}) = \mathsf{UR}_{\Delta_H(z)}(q) \tag{4}$$

$$\tilde{\mathcal{E}}_A^C(\pi) = \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q}) \tag{5}$$

Proof. We only prove (4) by induction here; (5) can be proved in a similar way. Let $\pi = \sigma_1\hat{\sigma}_{a1}\cdots\sigma_n\hat{\sigma}_{an}$ and $(q^i, \tilde{q}^i, z^i) = f(\pi_{[2i]})$ for $0 \le i \le n$. By the definition of AAS, when $z^i \neq z_{att}$ we have $z^i = \xi(\pi_{[2i]})$, which means that $\Delta_H(z^i) = S(\pi_{[2i]})$.

For the case of induction basis, i.e., $i = 0$, we have $\pi_{[0]} = \epsilon \in \mathcal{L}(S/G)$ and $\mathcal{E}_S^C(\pi_{[0]}) = \mathsf{UR}_{S(\epsilon)}(X_0) = \mathsf{UR}_{\Delta_H(z_0)}(q^0)$, which means (4) holds when $i = 0$. To proceed the induction, we assume that, for $0 \le i \le n-1$, (4) holds and $(\pi_{[2i]})_{\mathsf{S}} \in P(\mathcal{L}(S/G))$. For $j = i+1$, if $(\pi_{[2j]})_{\mathsf{S}} \in P(\mathcal{L}(S/G))$, then we have

$$\begin{aligned}\mathcal{E}_S^C(\pi_{[2j]}) &= \mathsf{UR}_{S(\pi_{[2j]})}(\mathsf{NX}_{\sigma_{aj}}(\mathcal{E}_S^C(\pi_{[2i]})))\\ &= \mathsf{UR}_{\Delta_H(z^j)}(\mathsf{NX}_{\sigma_{aj}}(\mathsf{UR}_{\Delta_H(z^i)}(q^i)))\\ &= \mathsf{UR}_{\Delta_H(z^j)}(q^j)\end{aligned}$$

which means (4) holds for $j = i+1$. If $(\pi_{[2j]})_{\mathsf{S}} \notin P(\mathcal{L}(S/G))$, then we know that $\sigma_{aj} \notin \mathcal{O}(\mathcal{E}_{S/G}^C(\pi_{[2i]}), S(\pi_{[2i]})) \cup \{\epsilon\} = \mathcal{O}(q^i, \Delta_H(z^i)) \cup \{\epsilon\}$. This implies that (i): $\hat{\sigma}_{aj} \neq \hat{\epsilon}$; and (ii) $\sigma_{aj} \notin \Delta_H(z^i)$ or $\mathsf{NX}_{\sigma_{aj}}(\mathsf{UR}_{\Delta_H(z^i)}(q^i)) = \emptyset$. Therefore $z_j = z_{\text{att}}$. By the definition of AAS, those environments with third components z_{att} have no successor. Hence, we have $j = n$ and $\mathsf{UR}_{\Delta_H(z^j)}(q^j) = \mathsf{UR}_{\Delta_H(z_{\text{att}})}(q^j) = \emptyset = \mathcal{E}_S^C(\pi_{[2j]})$, which also implies that (4) holds for $j = i+1$. This completes the inductive proof. □

Note that in the AAS M, the second component of an environment state represents the current-state estimate of the attacker over the augmented state-space $\tilde{G}$. Therefore, if we restrict to its IS-label component, we can determine whether or not the IS of the system is revealed to the intruder by: for any extended string $\pi \in \mathcal{L}_e(M)$ such that $(q, \tilde{q}, z) = f(\pi, y_0)$, we have

$$\mathcal{E}_A^I(\pi) \subseteq X_{sec} \Leftrightarrow I(\tilde{q}) = \{\mathbf{S}\} \tag{6}$$

$$\mathcal{E}_A^I(\pi) \cap X_{sec} = \emptyset \Leftrightarrow I(\tilde{q}) = \{\mathbf{NS}\} \tag{7}$$

Intuitively, the first equation corresponds to the case that the IS is detected by the intruder since all labels in the state set are secret, while the second equation corresponds to the case that the intruder knows for sure that the system was not from a secret initial state.

Also, we denote by $Y_{\text{att}} = \{(q, \tilde{q}, z) \in Y_e : z = z_{\text{att}}\}$ the set of *attack-expose* states, whose properties are characterized by the following result.

Proposition 2. *For any attacker A and $\pi \in \mathcal{L}_e(S_A/G)$, π_{A} is stealthy if and only if $f(y_0, \pi_{\mathsf{A}}) = (q, \tilde{q}, z) \notin Y_{\text{att}}$.*

Proof. Let $\pi_{\mathsf{A}} = \sigma_1\hat{\sigma}_{a1}\cdots\sigma_n\hat{\sigma}_{an}$. Since $\pi \in \mathcal{L}_e(S_A/G)$, we have $\sigma_i \in \mathcal{O}(\mathcal{E}_A^C((\pi_{\mathsf{A}})_{[2i-2]}), S((\pi_{\mathsf{S}})_{[2i-2]}))$ for $1 \le i \le n$, and $\hat{\sigma}_{ai} \in \mathcal{V}(\sigma_i)$. By the definition of AAS and Proposition 1, π_{A} is always well-defined in M, i.e., $\pi_{\mathsf{A}} \in \mathcal{L}_e(M)$, and $\mathcal{E}_S^C(\pi_{\mathsf{S}}) = \mathsf{UR}_{\Delta_H(z)}(q)$.

Then, for the "if" direction, when $f(y_0, \pi_{\mathsf{A}}) = (q, \tilde{q}, z) \notin Q_{\text{att}}$, $(q, \tilde{q}, z)$ has successors, i.e., $\mathsf{UR}_{\Delta_H(z)}(q) \neq \emptyset$. Therefore, $\mathcal{E}_S^C(\pi_{\mathsf{S}}) \neq \emptyset$, which means that $\pi_{\mathsf{S}} \in P(\mathcal{L}(S/G))$, i.e., π_{A} is stealthy. For the "only if" direction, when π_{A} is stealthy, $\pi_{\mathsf{S}} \in P(\mathcal{L}(S/G))$, which means $\mathcal{E}_S^C(\pi_{\mathsf{S}}) \neq \emptyset$. Then we know that $\mathsf{UR}_{\Delta_H(z)}(q) \neq \emptyset$. Since we have illustrated that for $\pi_{\mathsf{A}} \in \mathcal{L}_e(M)$, we have $z \neq z_{att}$ and as a result, $f(y_0, \pi_A) \notin Y_{att}$. □

We illustrate the AAS structure and its properties by the following example.

Example 4. Again, let us consider system G and supervisor S in Fig. 1 and we further assume that $\Sigma_v = \{o_1, o_2, o_3\}$ is the set of vulnerable events. Then (part of) the AAS is shown in Fig. 4, where rectangle nodes represent environment states, the circular nodes represent the attack states, solid arrows represent transitions labeled by actual events in Σ and dashed arrows represent transitions labeled by tampered events in $\hat{\Sigma}_o$. According to Definition 3, the initial state is $y_0 = (X_1, \tilde{X}_1, z_1)$, where X_1 are the initial states of the original system G, $\tilde{X}_1$ are the initial states of the augmented system $\tilde{G}$ and z_1 is the initial states of the supervisor automaton H. Then at state y_0, event in $\{o_1, o_2, o_3\} = \mathcal{O}_{\tilde{G}}(\tilde{X}_1, \Delta_H(z_1))$ may occur. If event o_1 occurs, the AAS will move to an attack state $y_1 = (X_1, \tilde{X}_1, z_1, o_1)$ by simply remembering o_1. Then at attack state y_1, since $o_1 \in \Sigma_v$, events in $\{\hat{\epsilon}, \hat{o}_1, \hat{o}_2, \hat{o}_3\}$ may be chosen. Suppose that this time the attacker chooses to erase o_1, i.e., event $\hat{\epsilon}$ is chosen, the system moves to a new environment state $y_2 = (X_1, \tilde{X}_2, z_1)$, where the first and the third components are the same with y_0 since the supervisor observes nothing. However, the second component, which is the (augmented) state estimate of the attacker, is updated to $\tilde{X}_2 = \widetilde{\mathsf{NX}}_{o_1}(\widetilde{\mathsf{UR}}_{\Delta_H(z_0)}(\tilde{X}_1)) = \{(\mathsf{S}, 4), (\mathsf{S}, 5), (\mathsf{NS}, 6)\}$. Again, at environment state y_2, events in $\{o_1, o_4\} = \mathcal{O}_{\tilde{G}}(\tilde{X}_2, \Delta_H(z_1))$ may occur. If event $o_4 \notin \Sigma_v$ occurs, then the AAS moves to attack state $y_3 = (y_2, o_4)$ at which the attacker has no choice but to keep observation o_4. Therefore, upon event $\hat{o}_4$, the AAE moves to environment state $y_4 = (\emptyset, \tilde{X}_1, z_{\text{att}})$, where the second component is updated by $\widetilde{\mathsf{NX}}_{o_4}(\widetilde{\mathsf{UR}}_{\Delta_H(z_1)}(\tilde{X}_2)) = \widetilde{\mathsf{NX}}_{o_4}(\widetilde{\mathsf{UR}}_{\Delta_H(z_1)}(\{(\mathsf{S}, 4), (\mathsf{S}, 5), (\mathsf{NS}, 6)\})) = \tilde{X}_1$. However, from the supervisor's point of view, we have $o_4 \notin \mathcal{O}(X_1, \Delta_H(z_1))$, i.e., it knows the presence of the attacker. Therefore, the first component is $\emptyset$ and the third component is z_{att}, i.e., y_4 is an attack-expose state.

5. The simplified AAS

In the previous section, we have introduced the AAS that embeds all attack strategies until their exposure. In practice, however, once the attacker knows for sure that the system was or was not initiated from secret states, its win or loss has been completely determined.

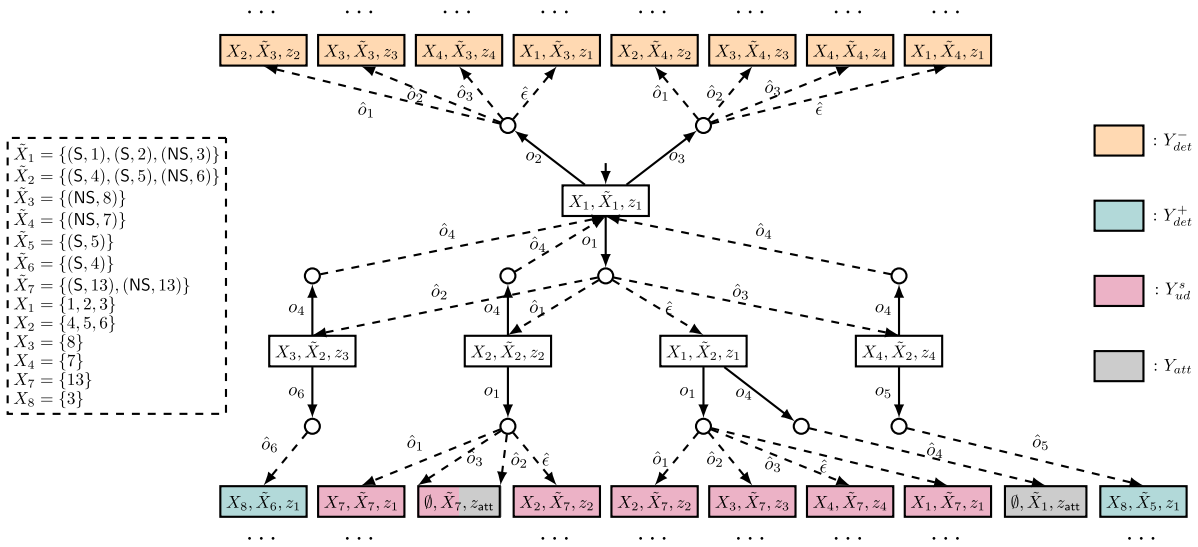


Fig. 4. The (part of) AAS M for system G and supervisor S in Fig. 1, where rectangle nodes represent the environment states in Y_e and circle nodes represent the attack states in Y_a. For the sake of simplicity, we do not show the state name of each attack state, since it can be uniquely determined by its predecessor environment state and its incoming transition.

Definition 4 (*Detected States*)**.** Given AAS M, for any environment state $y_e = (q, \tilde{q}, z) \in Q_e$, we say y_e is a

- *positive detected state* if $I(\tilde{q}) = \{\mathsf{S}\}$; and
- *negative detected state* if $I(\tilde{q}) = \{\mathsf{NS}\}$.

We denote by $Y^+_{det} \subseteq Y_e$ and $Y^-_{det} \subseteq Y_e$ the set of all positive and negative detected states, respectively, and define $Y_{det} = Y^+_{det} \cup Y^-_{det}$ as the set of detected states.

We observe that, once we know for sure that the system was or was not started from secret states, we know this information forever. In terms of the AAS, once all states in an environment state have the same label S or NS, all its successor environment states should also be so. This situation is summarized by the following result.

Proposition 3. *Let $y_e \in Y_e$ be a positive (respectively, negative) detected state. Then all environment states reachable from y_e are also positive (respectively, negative) detected states.*

Proof. Let $y_e = (q, \tilde{q}, z) \in Y_e$ be a positive (respectively, negative) detected state. For any transition $(y_e, o_1o_2, y'_e = (q', \tilde{q}', z')) \in f$, we have $\tilde{q}' = \widetilde{\mathsf{NX}}_{o_2}(\widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q}))$. Therefore, $I(\tilde{q}) = \{\mathbf{S}\}$ (respectively, $I(\tilde{q}) = \{\mathbf{NS}\}$) implies that $I(\tilde{q}') = \{\mathbf{S}\}$ (respectively, $I(\tilde{q}') = \{\mathbf{NS}\}$), which means that y'_e is also a positive (respectively, negative) detected state. By a recursive argument, we can conclude that all environment states reachable from y_e are also positive (respectively, negative) detected states. □

Detected states essentially provide a state-based winning condition for the attacker-player. On the other hand, we can also determine in advance that the attacker *cannot* win the game forever in some cases. This is captured by the notion of undetectable states.

Definition 5 (*Undetectable States*)**.** Given AAS M, for any environment state $y = (q, \tilde{q}, z) \in Y_e$, we say y is

- *weakly undetectable* if there exists a state $(\mathsf{S}, x) \in \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$ such that $(\mathsf{NS}, x) \in \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$; and
- *strongly undetectable* if $y \notin Y_{det}$ and for any state $(\mathsf{S}, x) \in \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$, we have $(\mathsf{NS}, x) \in \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$.

We denote by $Y^w_{ud} \subseteq Y_e$ and $Y^s_{ud} \subseteq Y_e$ the set of all weakly undetectable states and the set of all strongly undetectable states, respectively.

The intuitions of the weakly and strongly undetectable states are as follows. If the system under attack reaches a weakly undetectable state in the AAS, then it means that *there exist* two different strings in the system, one from a secret initial state and the other from a non-secret initial state, that end up with the same current-state x via strings having the same observation. Then if the actual string executed is the secret one, then the attacker will never be able to detect the IS in the future. This is because no matter how the secret string is expanded, there always exists the same expansion of the non-secret string. Therefore, once the system reaches a *weakly* undetectable state, we can conclude immediately that the attacker cannot be *strongly* IS-detectable.

On the other hand, if the system reaches a strongly undetectable state, it means that *for any* observation consistent string from a secret state, *there exists* another observation consistent string from a non-secret state such that they end up with the same current-state. Therefore, once the system reaches a *strongly* undetectable state, we can conclude immediately that the attacker cannot be *weakly* IS-detectable. Clearly, we have $Y^s_{ud} \subseteq Y^w_{ud}$.

The above observations are summarized by the following two propositions. First, we show that strongly undetectable states can be considered as "failure states" when considering the weak attacker synthesis problem.

Proposition 4. *Let $y \in Y_M$ be a strongly undetectable state. Then any environment state reachable from y is not a positive detected state.*

Proof. Let us consider an arbitrary environment states $y' = (q', \tilde{q}', z')$ reachable from $y = (q, \tilde{q}, z)$. If $I(\tilde{q}') = \{\mathsf{NS}\}$, then clearly it is negative detected. Else, for any $(\mathsf{S}, x') \in \widetilde{\mathsf{UR}}_{\Delta_H(z')}(\tilde{q}')$, since y' is reachable from y, there exists $(\mathsf{S}, x) \in \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$. However, since q is undetectable, we know that there exists $(\mathsf{NS}, x) \in \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$. This further implies that $(\mathsf{NS}, x') \in \widetilde{\mathsf{UR}}_{\Delta_H(z')}(\tilde{q}')$. Therefore, we conclude that y' is also undetectable. □

Based on Proposition 4, for a weakly IS-detectable attacker, once it reaches a strongly undetectable state, it will fail to reach

a positive detected state. Similarly, we show that weakly undetectable states can never be reached under a strongly IS-detectable attacker.

Proposition 5. *Suppose A is a strongly IS-detectable attacker. For any semi-stealthy extended string $\pi \in \mathcal{L}_e(S_A/G)$ such that $y = f(y_0, \pi_A)$, we have $y \notin Y^w_{ud}$.*

Proof. By contradiction. Assume that $y \in Y^w_{ud}$, i.e., there exists $(\mathsf{S}, x), (\mathsf{NS}, x) \in \widetilde{\mathrm{UR}}_{\Delta_H(z)}(\tilde{q})$. Let $\pi_1 \in \mathcal{L}_e(S_A/G, x_{0,1})$, $\pi_2 \in \mathcal{L}_e(S_A/G, x_{0,2})$ be two extended strings such that $x_{0,1} \in X_{sec}, x_{0,2} \notin X_{sec}$, $\delta((\pi_1)_G) = \delta((\pi_2)_G) = x, (\pi_1)_A = (\pi_2)_A$ and $f((\pi_2)_A) = y$. Then we have $\mathcal{L}(S_A/G, x_{0,1})/\pi_1 = \mathcal{L}(S_A/G, x_{0,2})/\pi_2$. Therefore, for all $\pi' \in \mathcal{L}(S_A/G, x_{0,1})/\pi_1$, we have $\{x_{0,1}, x_{0,2}\} \subseteq \mathcal{E}^I_A(P_A(\pi_1\pi'))$, which contradicts to the fact that the attacker is strongly IS-detectable. □

Based on Propositions 3, 4 and 5, we know that there is no need to construct the entire AAS for the purpose of strong or weak attacker synthesis. Specifically, once we reach a detected state in Y_{det}, the attacker knows for sure whether the initial state is secret or not and there is no need to expand the AAS anymore. Furthermore, for the purpose of synthesizing weakly (respectively, strongly) IS-detectable attackers, there is no need to further expand the AAS once it reaches a strongly (respectively, weakly) undetectable state since we know that the objective, i.e., surely (possibly) detect the IS cannot be achieved anymore.

To this end, we denote by

$$M^w = (Y^w, \Sigma_M, f^w, y_0),$$

the *weakly simplified AAS* (AAS-weak), which is the reachable part of the AAS M after removing all outgoing transitions from states $Y_{det} \cup Y^s_{ud}$. Similarly, we denote by

$$M^s = (Y^s, \Sigma_M, f^s, y_0),$$

the *strongly simplified AAS* (AAS-strong), which is the reachable part of the AAS M after removing all outgoing transitions from states $Y_{det} \cup Y^w_{ud}$.

Example 5. Let us still consider the AAS in Example 4. Negative/positive detected states and weakly/strongly undetectable states have been marked by different colors in Fig. 4. For example, since $I(\tilde{X}_3) = I(\tilde{X}_4) = \{\mathsf{NS}\}$, environment states with second components $\tilde{X}_3$ or $\tilde{X}_4$ are negative detected states. Also, since $I(\tilde{X}_5) = I(\tilde{X}_6) = \{\mathsf{S}\}$, environment states with second components $\tilde{X}_5$ or $\tilde{X}_6$ are positive detected states. And since $\tilde{X}_7 = \{(\mathsf{NS}, 13), (\mathsf{S}, 13)\}$, environment states with second components $\tilde{X}_7$ are strongly undetectable states (hence, also weakly undetectable states). For this example, since all weakly undetectable states are also strongly undetectable, the AAS-strong M^s and the AAS-weak M^w are the same, which is the part shown in Example 4.

Finally, we discuss the construction and the complexity of the (simplified) AAS. Specifically, the simplified AAS can be constructed by either a depth-first search or a breadth-first search starting from the initial state y_0. One just needs to traverse the state-space by enumerating all feasible transitions and the search is terminated whenever an attack-expose, detected or undetectable state is reached. In the worst-case, both the AAS and the simplified ones contain $2^{|X|} \cdot 2^{|X_0|\times|X|} \cdot |Z|$ states and $2^{|X|} \cdot 2^{|X_0|\times|X|} \cdot |Z| \cdot |\Sigma_o|$ transitions, which is exponential with respect to the size of the system. However, it is widely recognized that such an exponential complexity in the worst-case scenario appears to be inevitable for the majority of partial-observation synthesis problems.

Table 1
The state sizes of AAS M, weakly simplified AAS M^w and strongly simplified AAS M^s for two systems adopted from the literature.

System in Figure 1 of Saboori and Hadjicostis (2013)				System in Figure 1 of Saboori and Hadjicostis (2011b)			
X_{sec}	$\lvert M\rvert$	$\lvert M^w\rvert$	$\lvert M^s\rvert$	X_{sec}	$\lvert M\rvert$	$\lvert M^w\rvert$	$\lvert M^s\rvert$
{0}	48	22	9	{0}	53	32	19
{1}	48	22	9	{1}	60	9	9
{2}	36	9	9	{2}	43	9	9
{3}	36	9	9	{3}	60	9	9
–	–	–	–	{4}	43	9	9

On the other hand, in practical scenarios, the worst-case upper-bound above is rarely achieved. To see this, as well as the state-space reduction by the simplified AASs, we adopt two examples from the literature. Specifically, we consider the system in Figure 1 of Saboori and Hadjicostis (2013) and the system in Figure 1 of Saboori and Hadjicostis (2011b), where initial-state opacity was investigated. For both systems, we consider several cases, where different states are set to be the initial secret-states. For each case, we use the supervisor synthesis procedure in Saboori and Hadjicostis (2011a) to obtain the maximally permissive initial-state opacity enforcing supervisor. Furthermore, we assume that all observable events are vulnerable. Then for each system and its supervisor, we construct its AAS, weak simplified AAS and strong simplified AAS; the numbers of states for each system are shown in Table 1. According to the experimental results, one can see that, for each case, (i) the overall size of the AAS is much smaller than its theoretical upper bound; and (ii) the simplified AASs provide considerable state-space reductions compared with the original AAS.

6. Synthesis of attack strategies

In this section, we discuss how to synthesize attack strategies that are weakly or strongly IS-detectable. First, we discuss how to decode a finite-memory attack strategy by the Attack Strategy Structure. Then we show that the weak attacker synthesis problem can be solved using deterministic strategies, while the strong attacker synthesis problems may require non-deterministic strategies.

6.1. Attack strategy structures

In general, an attack strategy $A : (\Sigma_o \hat{\Sigma}_o)^* \Sigma_o \to 2^{\hat{\Sigma}_o}$ is string-based. Since the set of extended strings may be infinite, a strategy may require infinite memory to realize. In terms of the AAS, at each attack state, the attacker may issue different decisions for different visits. Hereafter, we will restrict the strategy space to the *attack-state-based* (AS-based) strategies. That is, the decision of the attacker only depends on the attack state reached in the AAS. Such AS-based strategies can be encoded as a special sub-automaton of the AAS.

Definition 6 (*Attack Strategy Structure*)**.** Let $m = (Y^m = Y^m_a \cup Y^m_e, \Sigma_M, f^m, y_0)$ be a sub-automaton of the (simplified) AAS M. We say m is an *Attack Strategy Structure* of M if

(1) for any environment state $y \in Y^m_e$, all feasible observations are defined, i.e., $|\Delta_m(y)| = |\Delta_M(y)|$; and
(2) for any attack state $y \in Y^m_a$, the attacker has at least one choice, i.e., $|\Delta_m(y)| > 0$.

We denote by $\mathbb{A}(M)$ the set of all Attack Strategy Structures of the (simplified) AAS M.

Then given an ASS $m \in \mathbb{A}(M)$, we can induce an AS-based attack strategy, denoted by A_m, as follows. For any $\pi\sigma \in (\Sigma_o \hat{\Sigma}_o)^* \Sigma_o$, we have

$$A_m(\pi\sigma) = \begin{cases} \Delta_m(f^m(y_0, \pi\sigma)) & \text{if } f^m(y_0, \pi\sigma)! \\ \{\hat{\sigma}\} & \text{otherwise} \end{cases}.$$

That is, the attack decision $A_m(\pi\sigma)$ is the set of events defined at the attack state $f^m(y_0, \pi\sigma)$ in structure m. Note that, in general, A_m is non-deterministic since $\Delta_m(f^m(y_0, \pi\sigma))$ may not be a singleton. We say that an ASS $m \in \mathbb{A}(M)$ is *deterministic* if for any $y \in Y_a^m$, we have $|\Delta_m(y)| = 1$. Clearly, for a deterministic AAS m, its induced strategy is deterministic.

6.2. Synthesis of weak attack strategies

Now, we discuss how to solve Problem 1 for the case of weakly IS-detectable attacker. First, we restrict our attention to *deterministic and AS-based* strategies, which can be realized by deterministic ASS. The following theorem shows that, for a deterministic ASS, its induced strategy is weakly IS-detectable if and only if it contains a positive detected state.

Theorem 1. *Given a deterministic ASS $m \in \mathbb{A}(M^w)$, its induced strategy A_m is weakly IS-detectable if and only if $Y^m \cap Y^+_{det} \neq \emptyset$.*

Proof. The necessity is straightforward. If $Y^m \cap Y^+_{det} = \emptyset$, for any extended string $\pi \in \mathcal{L}_e(S_{A_m}/G)$, π_A is either not semi-stealthy, or not able to detect the IS. To see the sufficiency, suppose that the deterministic ASS m contains a positive detected state. Let $\pi \in \mathcal{L}_e(m)$ be an extended string such that $f^m(\pi) \in Y^+_{det}$. Let $\pi' \in \mathcal{L}_e(S_{A_m}/G)$ and $\pi'_A = \pi$. Clearly, A_m is stealthy along $\pi_{[|\pi|-2]}$. By Proposition 1 and the property of positive detected state, π' is semi-detected. □

The above result restricts the solution space to deterministic and AS-based strategies. In general, a strategy can be non-deterministic or string-based. However, the following result further shows that such a restriction is, in fact, without loss of generality for the purpose of solvability of the weak attacker synthesis problem.

Theorem 2. *For a DES under supervisory control S/G, there exists a weakly IS-detectable attack strategy if and only if there exists a weakly IS-detectable attack strategy that is deterministic and AS-based.*

Proof. The sufficiency is straightforward. To see the necessity, suppose that there exists a weakly IS-detectable attack strategy A, which may not be encoded as a deterministic ASS. Let $\pi \in \mathcal{L}_e(S_A/G)$ be a shortest extended sequence such that (i) π is semi-stealthy; and (ii) $\mathcal{E}^I_A(\pi_A) \subseteq X_{sec}$. Then π_A is a well-defined extended string in M^w, which means that $y = f^w(\pi_A) = (q, \tilde{q}, z)$ is an environment state in M^w. Then by Proposition 2, we have $I(\tilde{q}) = \{S\}$, i.e., $y \in Y^+_{det}$. Therefore there exists a deterministic ASS $m \in \mathbb{A}(M)$ containing y, and according to Theorem 1, attack strategy induced by m is naturally weakly IS-detectable. □

The above two theorems suggest immediately a sound and complete approach for synthesizing a weakly IS-detectable attack strategy as follows.

- First, build the AAS-weak M^w and check whether or not it contains a positive detected state;
- If so, we find an arbitrary deterministic ASS $m \in \mathbb{A}(M^w)$ in M^w such that a positive detected state is contained, and its induced strategy A_m is the solution;
- Otherwise, no weakly IS-detectable attacker exists.

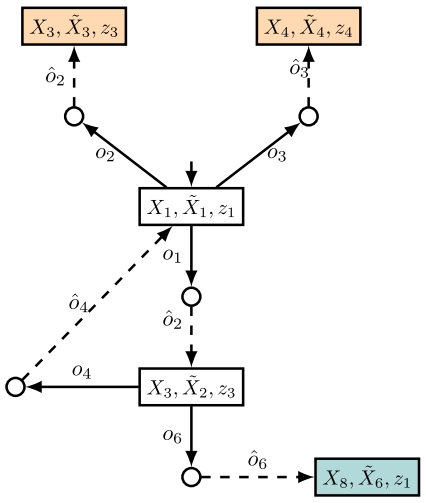


Fig. 5. The DASS of AAS-weak in Example 5 whose induced supervisor is weakly IS-detectable.

We illustrate the above process by the following example.

Example 6. Let us still consider the running example, where the AAS-weak is shown in Fig. 4. In the AAS-weak, since $(X_8, \tilde{X}_6, z_1)$, $(X_8, \tilde{X}_5, z_1) \in Y^+_{det}$, we can extract a deterministic ASS from AAS-weak such that it contains at least one positive detected state. AAS m shown in Fig. 5 is such a structure. By applying its induced attack strategy A_m, when the system generates o_1o_6, the attacker will replace the first o_1 by o_2, i.e., $o_1\hat{o}_2o_6\hat{o}_6 \in \mathcal{L}_e(S_{A_m}/G)$. Therefore, we have $\mathcal{E}^I_{S_{A_m}/G}(o_1\hat{o}_2o_6\hat{o}_6) \subseteq X_{sec}$ and A_m is weakly IS-detectable.

6.3. Synthesis of strong attack strategies

We consider AAS-strong for strong attack synthesis. In the previous subsection, we have shown that deterministic and AS-based strategies are sufficient for weakly IS-detectable attackers. However, the following example shows that such a solution space may not be sufficient for strongly IS-detectable attackers.

Example 7. We still consider our running example for which the AAS-strong is shown in Fig. 4. Let us investigate how the attacker makes a decision at attack state $y = (X_1, \tilde{X}_1, z_1, o_1)$, which is the circle state under the initial state. If the attacker follows a deterministic strategy, then it can replace o_1 either by $\hat{o}_2$ or by $\hat{o}_3$ deterministically. Note that it cannot choose $\hat{o}_1$ or $\hat{\epsilon}$ if it wants to enforce strong IS-detectability, since this will lead the system to weakly undetectable states. Suppose that the attack decision is $\{\hat{o}_2\}$. Then when the system is actually initiated from $2 \in X_{sec}$, the system under attack can only generate extended strings in $(o_1\hat{o}_2o_4\hat{o}_4)^*$ and the attacker can never detect the IS. Similarly, if the decision is $\{\hat{o}_3\}$, then the attacker can never detect the IS when the system is actually initiated from $1 \in X_{sec}$. Therefore, no deterministic and AS-based strategy exists for strong attackers.

However, one can easily find a *non-deterministic* but still AS-based strategy whose decision is $\{\hat{o}_2, \hat{o}_3\}$ at attack state y. That is, it will randomly replace o_1 by $\hat{o}_2$ or $\hat{o}_3$ in state y. Since y is in a loop and can be visited infinitely often before the detection of IS, no matter the system is initiated from $1 \in X_{sec}$ or $2 \in X_{sec}$, the IS will be detected almost surely as the system evolves.

The essential reason why deterministic strategies may not be sufficient is that, for different possible initial states, the attacker

may need to take different decisions to detect the IS. However, since the initial state is unknown, the attacker should not fix its decision; otherwise, it may block the opportunity for detecting other IS. In terms of the ASS, given an ASS m, it is not sufficient to check whether or not each state $y = (q, \tilde{q}, z)$ can reach a positive detected state for strong IS-detectability. In addition, we need to "zoom into" the state estimate $\tilde{q}$ and check whether or not each possible state $(\mathsf{S}, x) \in \tilde{q}$ in it can reach a positive detected state. This leads to the notion of *internal detectability*.

Definition 7 (*Internal-Detectability*)**.** Given an ASS $m \in \mathbb{A}(M^s)$, we say an environment state $y = (q, \tilde{q}, z) \in Y_e^m$ is *internally detectable* in m if for each $(\mathsf{S}, x) \in \widehat{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$, there exists a feasible extended string that can reach a positive detected state from y. That is, there exists $\pi = \sigma_1\hat{\sigma}_{a1}\cdots\sigma_n\hat{\sigma}_{an} \in (\Sigma_o\hat{\Sigma}_o)^*$ such that (i) $f^m(y, \pi) \in Y_{det}^+$; and (ii) there exists a sequence of states $x_0x_1\cdots x_n \in X^*$ such that $x_0 = x$ and for $i = 0, \ldots, n-1$, we have $x_{i+1} \in \mathsf{NX}_{\sigma_{i+1}}(\mathsf{UR}_{\Delta_H(z^i)}(x_i))$, where $(q^i, \tilde{q}^i, z^i) = f^m(y, \pi_{[2i]})$ is the ith environment state visited by π from y.

The intuitions of internal-detectability are as follows. Internal-detectability says that, if the system is actually initiated from a secret state and the IS is not yet detected, then it should be able to detect the IS along some extended string π. Note that, for different internal states $(\mathsf{S}, x), (\mathsf{S}, x') \in \widehat{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$, the extended strings for detecting the IS may be different. This is why deterministic ASS may not be sufficient. We denote by $Y_{int}^m \subseteq Y_e^m$ the set of all internal detectable states in m. Notice that, by definition, detected states Y_{det} are naturally internally detectable.

Now, given a (possibly non-deterministic) ASS m, in order to ensure that its induced strategy A_m is strongly IS-detectable, it suffices to require that all environment states in m are internally detectable. This property should hold globally for all reachable states; otherwise, it means that some possible IS may not be detected. This observation is formalized by the following key theorem, which shows that, to find a strong IS-detectable strategy, it suffices to find an ASS in which all states are internally detectable.

Theorem 3. *Given an ASS $m \in \mathbb{A}(M^s)$, its induced strategy A_m is strongly IS-detectable if and only if all environment states are internally detectable in m, i.e., $Y_{int}^m = Y_e^m$.*

Proof. We first prove the sufficiency. Assume that the system is started from a secret state $x_0 \in X_{sec}$ and generates an extended string $\pi \in \mathcal{L}_e(S_{A_m}/G, x_0)$. Let $y = (q, \tilde{q}, z) = f^m(\pi_\mathsf{A})$ and $x = \delta(x_0, \pi_\mathsf{G})$. Then we have $(\mathsf{S}, x) \in \widehat{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$. Since y is internally detectable, there exists an extended string $\pi' \in \mathcal{L}_e(S_{A_m}/G, x_0)/\pi$ such that $f^m(y, \pi'_\mathsf{A}) \in Y_{det}^+$. So $\pi\pi'$ is IS-detected. Since string π is arbitrary, we know that A_m is strongly IS-detectable.

Now we prove the necessity. For any environment state $y = (q, \tilde{q}, z)$ and any $(\mathsf{S}, x) \in \widehat{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$, there is a secret initial state $x_0 \in X_{sec}$ and extended string $\pi \in \mathcal{L}_e(S_{A_m}/G, x_0)$ such that π_A is semi-stealthy, $\delta(x_0, \pi_\mathsf{G}) = x$ and $f^m(y_0, \pi_\mathsf{A}) = y$. Since A_m is strongly IS-detectable, there exists $\pi\pi' \in \mathcal{L}_e(S_{A_m}/G, x_0)$ such that $\pi\pi'$ is IS-detected. Then we have $\pi_\mathsf{A}\pi'_\mathsf{A} \in \mathcal{L}_e(m)$ and $f^m(y_0, \pi_\mathsf{A}\pi'_\mathsf{A}) = f^m(y, \pi'_\mathsf{A}) \in Y_{det}^+$. Let π_i be a prefix of π' such that $(\pi_i)_\mathsf{A} = (\pi'_\mathsf{A})_{[2i]}$ and $x_i = \delta(x, (\pi_i)_\mathsf{G})$. Then the state sequence $x_0\cdots x_{|\pi'_\mathsf{A}|/2}$ satisfies the condition (ii) in Definition 7, which means that y is internally detectable. □

Then based on the above theorem, we propose Algorithm 1 to synthesize an ASS whose induced attacker is strongly IS-detectable. We denote the predecessor of states $Y' \subseteq Y$ in AAS M as $Pre(Y') = \{y \in Y : \exists\sigma \in \Sigma_M, f(y, \sigma) \in Y'\}$. The algorithm takes the AAS-strong M^s as the initial structure (line 1). Then for the current structure m, we need to check if all environment states are internally IS-detectable (line 2). If so, the structure satisfies the requirement in Theorem 3 and we output it as the solution. Otherwise, we need to remove those environment states that are not internally IS-detectable (lines 3, 5). Note that, by removing some environment states, some attack states may have no successor which violates the second condition of ASS. Therefore those attack states should be removed. Once those states are removed, some environment states may lose some successors which violates the first condition of ASS. Therefore those environment states should also be removed and new attack states need to be checked. The process is achieved by the inner while-loop (lines 4–12). The above process needs to be iterated since removing states may result in new environment states that are not internally IS-detectable. In brief, the outer while-loop enforces all environment states to be internally detectable and the inner while-loop enforces m to be an ASS. If the returned m is empty, the strong attack strategy does not exist. In fact, the ASS returned by the algorithm is the "largest" one among all AASs that induce strongly IS-detectable attackers. We illustrate the algorithm by the following example.

Algorithm 1: Strongly Attacker Synthesis

Input: $M^s = (Y^s = Y_e^s \cup Y_a^s, \Sigma_M, f^s, y_0)$
Output: ASS $m = (Y^m = Y_e^m \cup Y_a^m, \Sigma_M, f^m, y_0)$

1 $m \leftarrow M^s$;
2 **while** $Y_e^m \neq Y_{int}^m$ **do**
3 delete $Y_e^m \setminus Y_{int}^m$ from m;
4 **while** *exists* $y \in Y_a^m$ *s.t.* $\Delta_m(y) = \emptyset$ **do**
5 delete such attack state y and all its predecessor environment states $Pre(\{y\})$ from m;
6 **end**
7 $m \leftarrow$ the reachable part of m;
8 **end**
9 **return** *ASS* m

Example 8. We still consider the running example and the AAS-strong M^s shown in Fig. 4. Clearly, for the first execution of the outer while-loop, states in $Y_{ud}^s \cup Y_{att} \cup \{(X_1, \tilde{X}_2, z_1)\}$ are not internally IS-detectable since they cannot even reach a detected state in m. Therefore, we remove all these states from m. Now, we go to the inner while-loop and we find that attack state $(X_2, \tilde{X}_2, z_2, o_1)$ has no successor state. Therefore, we remove attack state $(X_2, \tilde{X}_2, z_2, o_1)$ as well as its predecessor environment state $(X_2, \tilde{X}_2, z_2)$. Then (the reachable part of) the resulting structure m after the inner while-loop is shown in Fig. 6. Now we see that all environment states in m are internally IS-detectable. Therefore, we jump out of the while-loop and terminate the algorithm. Then for the returned ASS m, its induced strongly IS-detectable attack strategy A_m is to replace o_1 at $(X_1, \tilde{X}_1, z_1)$ non-deterministically by either $\hat{o}_2$ or $\hat{o}_3$. As we have discussed in Example 7, this non-deterministic strategy is strongly IS-detectable.

The above synthesis algorithm essentially restricts our solution space to AS-based attack strategies. The following result shows that such a restriction is without loss of generality, i.e., the synthesis algorithm is not only sound but also complete.

Theorem 4. *For a DES under supervisory control S/G, there exists a strongly IS-detectable attack strategy if and only if there exists an AS-based one.*

Proof. The sufficiency is straightforward. To see the necessity, suppose that there exists a strongly IS-detectable attacker A, which may not be encoded as an ASS. Then we can construct a

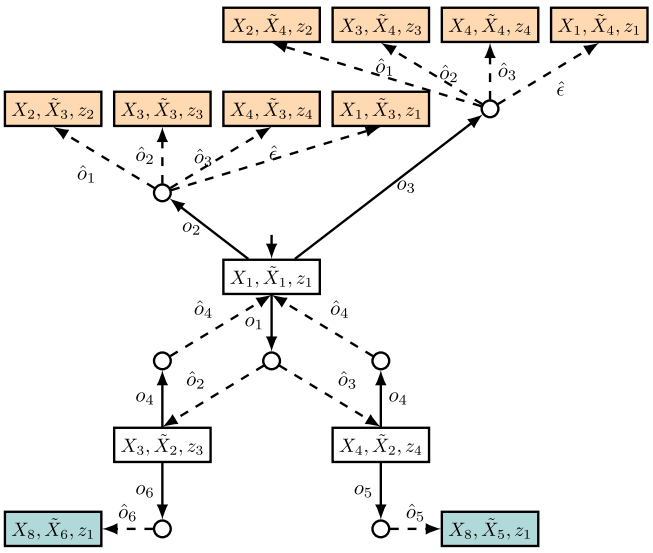


Fig. 6. The output ASS m whose induced supervisor is strongly IS-detectable.

new ASS m from A as follows. For any attack state $y \in Y_a$, its active events is defined by $\Delta_m(y) = \cup_{\pi\sigma\sigma' \in \mathcal{L}_e(S_A/G)_{\mathsf{A}}: f^s(\pi\sigma)=y} A(\pi\sigma)$, which is the union of all possible attack decisions of A when reaching y. We argue that, by constructing so, all environment states of the ASS m are internally IS-detectable. To see this, for any environment state $y = (q, \tilde{q}, z)$ in m and any (augmented) state $(\mathsf{S}, x) \in \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$, there exists an initial state $x_0 \in X_{sec}$ and an extended string $\pi \in \mathcal{L}_e(S_A/G, x_0)$ such that $f^s(\pi_{\mathsf{A}}) = y$ and $\delta(x_0, \pi_G) = x$. If the conditions in Definition 7 do not hold, then for any string $\pi' \in \mathcal{L}_e(S_A/G, x_0)/\pi$, we have $f^m(y, \pi'_{\mathsf{A}}) \notin Y^+_{det}$, which means that $\mathcal{E}^I_A(\pi_{\mathsf{A}}\pi'_{\mathsf{A}}) \not\subseteq X_{sec}$. This, however, contradicts to the fact that A is strong IS-detectable. Therefore, the constructed AS-based strategy A_m is also strongly IS-detectable according to Theorem 3. □

Remark 1. Note that, although a strongly IS-detectable attacker can guarantee the detection of the initial-secret almost surely, it does not exclude infinite strings along which the initial-secret is not detected. In other words, it cannot ensure that the initial-secret of the system can be revealed within a finite number of steps. The requirement of finite detection of initial-secret can be captured by the notion of *certain IS-detectability*. Formally, we say an attacker A is *certainly* IS-detectable if there exists a bound $n \in \mathbb{N}$ such that

$$(\forall x_0 \in X_{sec})(\forall \pi \in \mathcal{L}_e(S_A/G, x_0) : \pi \text{ is semi-stealthy})$$
$$[|\pi| \geq n \vee \pi \text{ is not stealthy}] \Rightarrow [\pi \text{ is IS-detected}] \quad (8)$$

In order to synthesize a certainly IS-detectable attacker, one can still use the AAS-strong M^s as the basis since this notion is even stronger than strong IS-detectability. However, instead of solving a non-blocking-like control problem over M^s, one needs to solve a *finite-reachability-like* control problem. Due to space constraints, a detailed synthesis algorithm for certainly IS-detectable attackers is beyond the scope of this work. In principle, it can be developed by combining the AAS-strong structure with the standard techniques for reachability games (Grädel, Thomas, & Wilke, 2002).

7. Conclusion

This paper investigated the problem of synthesizing active sensor attackers against initial-secret of supervisory control systems. Two levels of success of the attackers are considered, where strong IS-detectability requires that the attacker can almost surely detect the IS and weak IS-detectability requires that the attacker has the possibility of detecting the IS. To solve the synthesis problems, we defined an information-flow structure, called the AAS, that embeds all possible attack strategies. Using structural properties of initial-state estimation and initial-secret, we further simplify the AAS for attacker synthesis. Algorithms are presented for finding both weak attackers and strong attackers. We showed that the deterministic strategies are sufficient for weak attackers, while strong attackers may require non-deterministic strategies. This work follows a qualitative framework, where the attack objective is purely logic. In the future, we would like to investigate a quantitative setting by further taking attack costs into account.

References

Alves, M., Pena, P. N., & Rudie, K. (2021). Discrete-event systems subject to unknown sensor attacks. *Discrete Event Dynamic Systems*, 1–16.

Balun, J., & Masopust, T. (2021). Comparing the notions of opacity for discrete-event systems. *Discrete Event Dynamic Systems*, *31*(4), 553–582.

Basilio, J. C., Hadjicostis, C. N., & Su, R. (2021). Analysis and control for resilience of discrete event systems: Fault diagnosis, opacity and cyber security. *Foundations and Trends® in Systems and Control*, *8*(4), 285–443.

Bourouis, A., Klai, K., Ben Hadj-Alouane, N., & El Touati, Y. (2017). On the verification of opacity in web services and their composition. *IEEE Transactions on Services Computing*, *10*(1), 66–79.

Carvalho, L. K., Wu, Y.-C., Kwong, R., & Lafortune, S. (2018). Detection and mitigation of classes of attacks in supervisory control systems. *Automatica*, *97*, 121–133. http://dx.doi.org/10.1016/j.automatica.2018.07.017, https://www.sciencedirect.com/science/article/pii/S0005109818303741.

Cassandras, C. G., & Lafortune, S. (2021). *Introduction to discrete event systems: Vol. 3*, Springer.

Dubreil, J., Darondeau, P., & Marchand, H. (2010). Supervisory control for opacity. *IEEE Transactions on Automatic Control*, *55*(5), 1089–1100.

Fritz, R., & Zhang, P. (2018). Modeling and detection of cyber attacks on discrete event systems. *IFAC-PapersOnLine*, *51*(7), 285–290.

Grädel, Erich, Thomas, Wolfgang, & Wilke, Thomas (Eds.), (2002). *Automata logics, and infinite games: A guide to current research*. Berlin, Heidelberg: Springer-Verlag.

He, Z., Ma, Z., & Tang, W. (2021). Performance safety enforcement in strongly connected timed event graphs. *Automatica*, *128*, Article 109605.

Jacob, R., Lesage, J.-J., & Faure, J.-M. (2016). Overview of discrete event systems opacity: Models, validation, and quantification. *Annual Reviews in Control*, *41*, 135–146. http://dx.doi.org/10.1016/j.arcontrol.2016.04.015, https://www.sciencedirect.com/science/article/pii/S1367578816300189.

Khoumsi, A. (2019). Sensor and actuator attacks of cyber-physical systems: A study based on supervisory control of discrete event systems. In *8th International conference on systems and control* (pp. 176–182).

Lai, A., Lahaye, S., & Li, Z. (2021). Initial-state detectability and initial-state opacity of unambiguous weighted automata. *Automatica*, *127*, Article 109490.

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, *47*(3), 496–503.

Lin, L., & Su, R. (2021). Synthesis of covert actuator and sensor attackers. *Automatica*, *130*, Article 109714.

Lin, L., Tai, R., Zhu, Y., & Su, R. (2021). Heuristic synthesis of covert attackers against unknown supervisors. In *60th IEEE conference on decision and control* (pp. 7003–7008). IEEE.

Liu, S., Trivedi, A., Yin, X., & Zamani, M. (2022). Secure-by-construction synthesis of cyber-physical systems. *Annual Reviews in Control*.

Ma, Z., & Cai, K. (2022). Optimal secret protections in discrete-event systems. *IEEE Transactions on Automatic Control*.

Meira-Góes, R., Kang, E., Kwong, R. H., & Lafortune, S. (2020). Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems. *Automatica*, *121*, Article 109172.

Meira-Góes, R., Lafortune, S., & Marchand, H. (2021). Synthesis of supervisors robust against sensor deception attacks. *IEEE Transactions on Automatic Control*, *66*(10), 4990–4997.

Rashidinejad, A., Wetzels, B., Reniers, M., Lin, L., Zhu, Y., & Su, R. (2019). Supervisory control of discrete-event systems under attacks: An overview and outlook. In *18th European control conference* (pp. 1732–1739).

Saboori, A., & Hadjicostis, C. N. (2011a). Opacity-enforcing supervisory strategies via state estimator constructions. *IEEE Transactions on Automatic Control*, *57*(5), 1155–1165.

Saboori, A., & Hadjicostis, C. N. (2011b). Verification of infinite-step opacity and complexity considerations. *IEEE Transactions on Automatic Control*, *57*(5), 1265–1269.

Saboori, A., & Hadjicostis, C. N. (2013). Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, *246*, 115–132.
Su, R. (2018). Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations. *Automatica*, *94*, 35–44.
Tong, Y., Li, Z., Seatzu, C., & Giua, A. (2018). Current-state opacity enforcement in discrete event systems under incomparable observations. *Discrete Event Dynamic Systems*, *28*(2), 161–182.
Wakaiki, M., Tabuada, P., & Hespanha, J. (2019). Supervisory control of discrete-event systems under attacks. *Dynamic Games and Applications*, *9*(4), 965–983.
Wang, Y., Li, Y., Yu, Z., Wu, N., & Li, Z. (2021). Supervisory control of discrete-event systems under external attacks. *Information Sciences*, *562*, 398–413.
Wintenberg, A., Blischke, M., Lafortune, S., & Ozay, N. (2022). A general language-based framework for specifying and verifying notions of opacity. *Discrete Event Dynamic Systems*, 1–37.
Wu, Y.-C., & Lafortune, S. (2013). Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, *23*(3), 307–339.
Xie, Y., Yin, X., & Li, S. (2021). Opacity enforcing supervisory control using non-deterministic supervisors. *IEEE Transactions on Automatic Control*.
Yang, S., Yin, X., Li, S., & Zamani, M. (2020). Secure-by-construction optimal path planning for linear temporal logic tasks. In *59th IEEE conference on decision and control* (pp. 4460–4466). IEEE.
Yao, J., Yin, X., & Li, S. (2022). Sensor deception attacks against initial-state privacy in supervisory control systems. In *61th IEEE conference on decision and control* (pp. 4839–4845).
Yin, X., & Lafortune, S. (2016). A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, *61*, 2140–2154. http://dx.doi.org/10.1109/TAC.2015.2484359.
Yu, X., Yin, X., Li, S., & Li, Z. (2022). Security-preserving multi-agent coordination for complex temporal logic tasks. *Control Engineering Practice*, *123*, Article 105130.
Zeng, W., & Koutny, M. (2021). Quantitative analysis of opacity in cloud computing systems. *IEEE Transactions on Cloud Computing*, *9*(3), 1210–1219.
Zhang, Q., Li, Z., Seatzu, C., & Giua, A. (2018). Stealthy attacks for partially-observed discrete event systems. In *ETFA* (pp. 1161–1164).

Jingshi Yao was born in Hunan, China, in 1999. She received her B.S. degree and M.S. degree from Shanghai Jiao Tong University in 2020 and 2023, respectively. She is pursuing the Ph.D. degree in the School of Chemistry, Chemical Engineering and Biotechnology at Nanyang Technological University. Her research interests lie in the security of cyber–physical systems using formal methods and distributed reinforcement learning for complex systems.

Shaoyuan Li was born in Hebei, China, in 1965. He received the B.S. and M.S. degrees in automation from the Hebei University of Technology, Tianjin, China, in 1987 and 1992, respectively, and the Ph.D. degree from Nankai University, Tianjin, in 1997. Since 1997, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently a chair Professor. His current research interests include model predictive control, dynamic system optimization, and cyber–physical systems. He is the vice-president of the Chinese Association of Automation.

Xiang Yin was born in Anhui, China, in 1991. He received the B.Eng degree from Zhejiang University in 2012, the M.S. degree from the University of Michigan, Ann Arbor, in 2013, and the Ph.D. degree from the University of Michigan, Ann Arbor, in 2017, all in electrical engineering. Since 2017, he has been with the Department of Automation, Shanghai Jiao Tong University, where he is an Associate Professor. His research interests include formal methods, discrete-event systems and cyber–physical systems. Dr. Yin is serving as the chair of the *IEEE CSS Technical Committee on Discrete Event Systems*, Associate Editors for the *Journal of Discrete Event Dynamic Systems: Theory & Applications*, the *IEEE Control Systems Letters*, the *IEEE Transactions on Intelligent Vehicles*, and a member of the *IEEE CSS Conference Editorial Board*. Dr. Yin received the IEEE Conference on Decision and Control (CDC) Best Student Paper Award Finalist in 2016.