2021 61st IEEE Conference on Decision and Control (CDC)
December 6-9, 2022. Cancún, Mexico

Sensor Deception Attacks Against Initial-State Privacy in Supervisory Control Systems

Jingshi Yao, Xiang Yin and Shaoyuan Li

Abstract— **This paper investigates the problem of synthesizing sensor deception attackers against privacy in the context of supervisory control of discrete-event systems (DES). We consider a plant controlled by a supervisor, which is subject to sensor deception attacks. Specifically, we consider an *active attacker* that can tamper with the observations received by the supervisor. The privacy requirement of the supervisory control system is to maintain *initial-state opacity*, i.e., it does not want to reveal the fact that it was initiated from a secret state during its operation. On the other hand, the attacker aims to deceive the supervisor, by tampering with its observations, such that initial-state opacity is violated due to incorrect control actions. We investigate from the attacker's point of view by presenting an effective approach for synthesizing sensor attack strategies threatening the privacy of the system. To this end, we propose the All Attack Structure (AAS) that records state estimates for both the supervisor and the attacker. This structure serves as a basis for synthesizing a sensor attack strategy. We also discuss how to simplify the synthesis complexity by leveraging the structural properties. A running academic example is provided to illustrate the synthesis procedure.**

I. Introduction

With the developments of computation and communication technologies, cyber-physical system (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 exchange information in real time. This, on the one hand, enables more flexible and intelligent control architectures, but on the other hand, makes security and privacy considerations become increasingly more important in the analysis and design of CPS [1]–[4].

In this paper, we study privacy issues in CPS whose high-level behaviors are abstracted as discrete-event systems (DES). Specifically, we focus on an important class of information-flow property called opacity, which captures whether or not some "secret" of the system can be inferred by an outsider via its observation. 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 [5]–[11]. In this work, we will focus on the notion of *initial-state opacity* [12], which requires that the system can never reveal the fact that it was initiated from a secret state.

This work was supported by the National Natural Science Foundation of China (62061136004, 61803259, 61833012) and the National Key Research and Development Program of China (2018AAA0101700).

Jingshi Yao, Xiang Yin and Shaoyuan Li are with Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China. {yaojingshi,yinxiang,syli}@sjtu.edu.cn.

In general, an uncontrolled open-loop system may not be opaque naturally. Therefore, *supervisors* are usually used to restrict the behavior of the system such that the closed-loop system under control is opaque. In the literature on DES, many algorithms have been developed for designing opacity-enforcing supervisors; see, e.g., [13]–[17]. All existing opacity-enforcing supervisory control systems are designed for the nominal setting in the sense that no malicious attacks exist. However, in networked environments, supervisors are usually subject to active and malicious attacks, which may destroy their desired closed-loop property in the nominal setting. Particularly, the sensors of the supervisor may be compromised under *sensor deception attacks* [18]–[23] such that the supervisor may receive factitious observation tampered with by the attacker. Then based on the incorrect information, the supervisor can be misled to take incorrect actions, which may further expose 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 supervisor, 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 may mislead the supervisor to expose its secret initial-state, i.e., to violate initial-state opacity; and (ii) at the same time, maintain itself undetected by the supervisor. To solve this problem, we build an information 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 an attacker strategy satisfying the above two requirements.

Our work is most related to [21], [22], where the authors also investigate how to synthesize sensor deception attacks against given supervisory control systems. However, the attack objective considered therein is the violation of safety, which is defined on the actual behavior of the system. Here, we consider the violation of privacy as the attack objective, which is a property defined in the information flow. To our knowledge, the synthesis of active attackers against privacy requirements has not yet been studied in the context of supervisory control of DES.

II. Preliminary

A. Supervisory Control of DES

Let Σ be a finite set of events. A string over Σ is a finite sequence $s = \sigma_1 \cdots \sigma_n, \sigma_i \in \Sigma$; $|s| = n$ denotes its length;

$s^i = \sigma_1 \cdots \sigma_i$ denotes the prefix of s with length $i \leq n$ and $s^0 = \epsilon$ is the empty string. We denote by Σ^* the set of all strings over Σ including the empty string ϵ. A language $L \subseteq \Sigma^*$ is a set of strings We define $\Sigma^\epsilon = \Sigma \cup \{\epsilon\}$.

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

In the context of supervisory control, the event set Σ is partitioned as $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo} = \Sigma_c \dot{\cup} \Sigma_{uc}$, 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. We define $P : \Sigma^* \to \Sigma_o^*$ as natural projection that replaces unobservable events in a string by ϵ, which can also be extended to $P : 2^{\Sigma^*} \to 2^{\Sigma_o^*}$ by $P(L) = \{P(s) \in \Sigma_o^* : s \in L\}$.

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 *control decisions* or control patterns. That is, upon the occurrence of $\alpha \in \Sigma_o^*$, events in $S(\alpha) \in \Gamma$ are *enabled* by the supervisor. 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 as:

- $\epsilon \in \mathcal{L}(S/G, x_0)$; 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 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 supervisor S is *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., [24].

Example 1. Let us consider system G shown in Figure 1(a), where $\Sigma_o = \{b, c, d\}$ and $\Sigma_c = \{a, c, d\}$. Let us consider a supervisor S that disables event c only when observing string b and enables all events otherwise. Then supervisor S can be realized by automaton H shown in Figure 1(b). The closed-loop system under control S/G is recognized by automaton $G \times H$ as Figure 2 shows.

Let $q \subseteq X$ be a set of states, $\gamma \subseteq \Sigma$ be a set of events and $\sigma \in \Sigma_o$ be an observable event. The *unobservable reach* of q under event set γ is $\mathsf{UR}_\gamma(q) := \{\delta(x, s) \in X : x \in q, s \in (\Sigma_{uo} \cap \gamma)^*\}$. The *observable reach* of q upon the occurrence of σ is $\mathsf{NX}_\sigma(q) := \{\delta(x, \sigma) \in X : x \in q\}$. We also define $\mathsf{NX}_\epsilon(q) = q$ for technical reason. The set of events that can be observed from q under event set γ is denoted as:

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

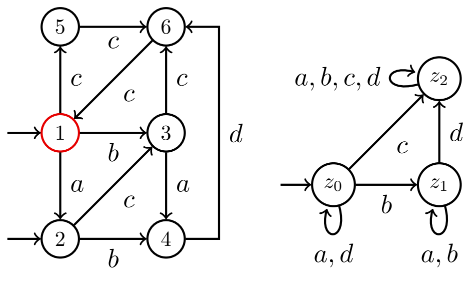

Fig. 1. System G and supervisor H, where $\Sigma_o = \{b, c, d\}$, $\Sigma_c = \{a, c, d\}$. The secret initial state $X_{sec} = \{1\}$ is marked by red circle.

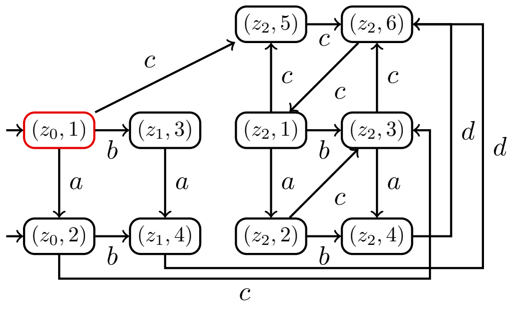

Fig. 2. Closed-loop system $G \times H$ in Example 1.

B. Initial-State and Current-State Estimates

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 $\mathcal{E}^C_{S/G}(\alpha)$ and $\mathcal{E}^I_{S/G}(\alpha)$, respectively, the current-state estimate and the initial-state estimate of the closed-loop system S/G upon observing α, i.e.,

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

Using the current-state estimate, for closed-loop system S/G, we have $\alpha\sigma \in P(\mathcal{L}(S/G))$ iff $[\alpha \in P(\mathcal{L}(S/G))] \wedge [\sigma \in \mathcal{O}(\mathcal{E}^C_{S/G}(\alpha), S(\alpha))]$.

Note that the current-state estimate $\mathcal{E}^C_{S/G}(\alpha)$ can be computed recursively by:

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

The initial-state estimate $\mathcal{E}^I_{S/G}(\alpha)$ can be computed through the current-state estimate over the augmented state-space [25]. Specifically, the *augmented system* of G is an automaton $\tilde{G} = (\tilde{X}, \Sigma, \tilde{\delta}, \tilde{X}_0)$, where $\tilde{X} \subseteq X_0 \times X$ is the set of states, $\tilde{X}_0 = \{(x_0, x_0) \in \tilde{X} : x_0 \in X_0\}$ is the set of initial-states, and the transition function $\tilde{\delta} : \tilde{X} \times \Sigma \to \tilde{X}$ is defined by: for any $(x_0, x) \in \tilde{X}, \sigma \in \Sigma$, we have $\tilde{\delta}((x_0, x), \sigma) =$

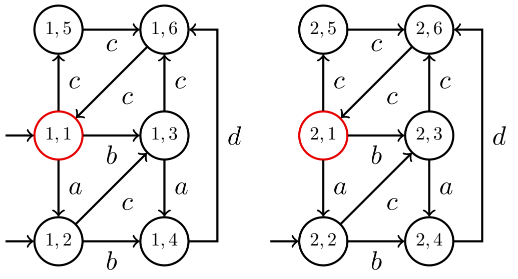

Fig. 3. Augmented system $\tilde{G}$ for system G in Figure 1(a), where red-marked states $\{(1,1),(2,1)\}$ correspond to the original secret state 1 in G.

$(x_0, \delta(x,\sigma))$. Then we have $\mathcal{L}(\tilde{G}) = \mathcal{L}(G)$ and $\mathcal{L}(S/\tilde{G}) = \mathcal{L}(S/G)$. Furthermore, for any set of augmented states $\tilde{q} \subseteq \tilde{X}$, we define $I(\tilde{q}) = \{x_0 \in X_0 : (x_0, x) \in q\}$ as the set of its first components. Then for any observation $\alpha \in P(\mathcal{L}(S/G))$, we have

$$\mathcal{E}^I_{S/G}(\alpha) = I(\mathcal{E}^C_{S/\tilde{G}}(\alpha)). \quad (1)$$

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

Example 2. Still, we consider system G and supervisor H shown in Figure 1. Suppose that string $\alpha = b \in P(\mathcal{L}(S/G))$ is observed. We have $\mathcal{E}^C_{S/G}(\alpha) = \{3,4\}$ and $\mathcal{E}^I_{S/G}(\alpha) = \{1,2\}$. Specifically, $\mathcal{E}^I_{S/G}(\alpha)$ can also be computed as follows. First, we construct the augmented system $\tilde{G}$ shown in Figure 3. Then we have $\mathcal{E}^C_{S/\tilde{G}}(\epsilon) = \mathsf{UR}_{S(\epsilon)}(\tilde{X}_0) = \{(1,1),(1,2),(2,1),(2,2)\}$ and $\mathcal{E}^C_{S/\tilde{G}}(b) = \mathsf{UR}_{S(b)}(\mathsf{NX}_b(\mathcal{E}^C_{S/\tilde{G}}(\epsilon))) = \{(1,3),(1,4),(2,3),(2,4)\}$. Finally, we get $\mathcal{E}^I_{S/G}(b) = I(\mathcal{E}^C_{S/\tilde{G}}(b)) = \{1,2\}$.

C. Initial-State Opacity

In some cases, for example, due to insecure communications, the observation of the system may also be available to an outsider. Here, we first consider a passive intruder (eavesdropper) 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 that of the supervisor.

Furthermore, we assume that the system has some "secret" that should not be revealed to the outside world. In this work, we assume that the system wants to hide the fact that it starts from some secret initial states $X_{sec} \subseteq X_0$. This requirement can be formalized by the notion of *initial-state opacity* defined as follows.

Definition 1 (Initial-State Opacity)**.** Given system G, observable events Σ_o, supervisor S and a set of secret initial states $X_{sec} \subseteq X_0$, the closed-loop system S/G is said to be initial-state opaque w.r.t. X_{sec} and Σ_o if

$$(\forall x_0 \in X_{sec})(\forall s \in \mathcal{L}(S/G, x_0))$$
$$(\exists x_0' \in X_0 \setminus X_{sec})(\exists t \in \mathcal{L}(S/G, x_0'))[P(s) = P(t)] \quad (2)$$

or equivalently,

$$(\forall \alpha \in P(\mathcal{L}(S/G)))[\mathcal{E}^I_{S/G}(\alpha) \not\subseteq X_{sec}]. \quad (3)$$

In practice, the open-loop system may not be initial-state opaque without control. Therefore, supervisor S is usually designed to guarantee that the closed-loop system S/G is initial-state opaque; see, e.g., [13]–[17] for how to synthesize such a supervisor. We illustrate the opacity-enforcing supervisor by the following example.

Example 3. We still consider system G shown in Figure 1(a) and assume state 1 is a secret initial-state, i.e., $X_{sec} = \{1\}$. Specifically, if the system is initiated from 1, then when sequence $bc \in P(\mathcal{L}(G,1)) \setminus P(\mathcal{L}(G,2))$ is observed, we have $\mathcal{E}^I_{S/G}(bc) = \{1\} \subseteq X_{sec}$, i.e., the intruder knows for sure that the system was from a secret-state. However, under the supervision of S shown in Figure 1(b), the closed-loop system under control shown as Figure 2 is initial-state opaque because the secret-revealing string bc is no longer in the closed-loop language $\mathcal{L}(S/G)$.

III. Active Attack Against Initial-State Opacity

A. Motivating Example

In the standard problem of initial-state opacity, 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 3, 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.

However, in networked control systems, some powerful attackers may further have the capability to *actively tamper* with the observations sent by the sensors in order to mislead the feedback decisions. Such an attacker model is referred to as the *sensor deception attacks* [21], [22]. The following example shows that, in the case of sensor deception attacks, the attacker may actively mislead a supervisor, which is original opaque-enforcing, to expose its secret initial state.

Example 4. Let us still consider system G and supervisor S in Figure 1. Now, we consider an active sensor attacker A described above. Suppose that the attacker A can *erase* the first occurrence of observable event b, e.g., by manipulating the communication channels between the sensors and the supervisor. Therefore, when observable string $b\beta$ actually occurs, the supervisor will only receive β. Recall that supervisor S simply disables event c when it observes string b. Therefore, under attacker A, string bc is still feasible since the supervisor does not know that event b happens. Then when the attacker observes bc, it knows for sure that the system was initiated from secret state 1.

B. Active Attackers and Problem Formulation

To formulate the above setting, we define $\Sigma_v \subseteq \Sigma_o$ as the set of *vulnerable events* whose occurrences can potentially be tampered with by an *active attacker*. An active attacker (or attack strategy) is a function

$$A : P(\mathcal{L}(G)) \to \Sigma_o^\epsilon,$$

satisfying the following constraints:

- $A(\epsilon)=\epsilon$;
- for any $\alpha\sigma\in P(\mathcal{L}(G))$, $A(\alpha\sigma)\in\begin{cases}\{\sigma\} & \text{if } \sigma\notin\Sigma_v\\ \Sigma_v^\epsilon & \text{if } \sigma\in\Sigma_v\end{cases}$.

That is, upon the occurrence of each new observable event $\sigma\in\Sigma_o$, if it is a vulnerable event, then the intruder may choose to either erase this event or replace it with another event in Σ_v. An active attacker A essentially induces a modification mapping for observable strings $g_A: P(\mathcal{L}(G))\rightarrow\Sigma_o^*$, which is defined recursively as:

- $g_A(\epsilon)=A(\epsilon)$; and
- $\forall\alpha\sigma\in P(\mathcal{L}(G)): g_A(\alpha\sigma)=g_A(\alpha)A(\alpha\sigma)$.

Therefore, upon the occurrence of $s\in\mathcal{L}(G)$, the supervisor will observe $g_A(P(s))$ and issue decision $S(g_A(P(s)))$. Essentially, system G can be regarded as being controlled by a new "supervisor" $S_A=S\circ g_A$, where $\circ$ is the standard function composition. Then the language generated by the controlled system under attack is given by $\mathcal{L}(S_A/G)$.

Before the supervisor detect the presence of the attacker, it will estimate the state based on the doctored observation in original closed-loop system S/G since it is not aware of the attacker. On the other hand, the attacker will estimate the state based on the real observation in attacked closed-loop system S_A/G. In summary, upon the occurrence of actual observation $\alpha\in P(\mathcal{L}(S_A/G))$,

- from the supervisor's point of view, the current-state estimate and the initial-state estimate are, respectively, $\mathcal{E}^C_{S/G}(g_A(\alpha))$ and $\mathcal{E}^I_{S/G}(g_A(\alpha))$;
- from the attacker's point of view, the current-state estimate and the initial-state estimate are, respectively, $\mathcal{E}^C_{S_A/G}(\alpha)$ and $\mathcal{E}^I_{S_A/G}(\alpha)$.

Note that, for actual observation $\alpha\in P(\mathcal{L}(S_A/G))$, if the attacker changes it to observation $g_A(\alpha)\notin P(\mathcal{L}(S/G))$, or equivalently, $\mathcal{E}^C_{S/G}(g_A(\alpha))=\mathcal{E}^I_{S/G}(g_A(\alpha))=\emptyset$, then the supervisor will detect the presence of the attacker and then, it may upgrade the security level or take further actions to protect the system from being attacked. Formally, we say attack A is *stealthy* along sequence $\alpha\in P(\mathcal{L}(S_A/G))$ if

$$\mathcal{E}^C_{S/G}(g_A(\alpha))\neq\emptyset. \tag{4}$$

In this work, we investigate from the attacker's point of view. The objective of the attacker is to break the privacy guarantee of the system, in the sense that the initial-secret (IS) may possibly be revealed under attack.

Problem 1 (IS-Detectable Attacker Synthesis Problem)**.** Given system G, observable events $\Sigma_o\subseteq\Sigma$, supervisor S, secret initial-states $X_{sec}\subseteq X_0$ and vulnerable events $\Sigma_v\subseteq\Sigma_o$, synthesize an attacker $A: P(\mathcal{L}(G))\rightarrow\Sigma_o^\epsilon$ such that $\exists\alpha\sigma\in\mathcal{L}(S_A/G)$ satisfying:

1) A is stealthy along α;
2) $\mathcal{E}^I_{S_A/G}(\alpha\sigma)\subseteq X_{sec}$.

If the attacker exists, we said the system S/G to be IS-attackable, and the attacker A to be IS-detectable.

Intuitively, in the above formulation, $\alpha\sigma$ is a string along which the attacker can detect the secret initial state, and the supervisor should not be aware of its presence before it successfully does so. Here, we only require the *existence* of such an attack string since we consider synthesizing an attacker that can *potentially threaten* the privacy of the system. As a stronger requirement, one may also require that the attacker can *always* detect the secret; this problem, however, is beyond scope of this work.

IV. 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 be served as the basis for our synthesis problem.

We denote by $\hat{\Sigma}^\epsilon=\{\hat{\sigma}:\sigma\in\Sigma\}\cup\{\hat{\epsilon}\}$ the events modified by the attacker in order to distinguish from the actually events generated by the plant. Note that the supervisor cannot distinguish event σ and $\hat{\sigma}$. For every observable event $\sigma\in\Sigma_o$, we define the action space of the attacker as

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

Definition 2 (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 finite-state automaton

$$M=(Q,\Sigma_M,f,q_0), \tag{5}$$

where

- $Q=Q_e\dot{\cup}Q_a$ such that $Q_e\subseteq 2^X\times 2^{\tilde{X}}\times(Z\cup\{z_{\text{att}}\})$ is the set of *environment states*, where z_{att} is a new state meaning that the supervisor has realize the presence of the attacker and $Q_a\subseteq 2^X\times 2^{\tilde{X}}\times Z\times\Sigma_o$ is the set of *attack states*;
- $q_0=(X_0,\tilde{X}_0,z_0)\in Q_e$ is the initial state;
- $\Sigma_M=\Sigma_o\cup\hat{\Sigma}^\epsilon_o$ is the set of events;
- $f: Q\times\Sigma_M\rightarrow Q$ is the transition function consists of the following two types of transitions:
  - $f_{ea}: Q_e\times\Sigma_o\rightarrow Q_a$ is the transition function from environment states to attack states defined by: for any $q_e=(q,\tilde{q},z)\in Q_e$, we have
  $$\Delta_M(q_e)=\begin{cases}\mathcal{O}(\tilde{q},\Delta_H(z)) & \text{if } z\in Z\\ \emptyset & \text{if } z=z_{\text{att}}\end{cases}.$$
  and for $\sigma\in\Delta_M(q_e)$, we have $f_{ea}(q_e,\sigma)=(q,\tilde{q},z,\sigma)$.
  - $f_{ae}: Q_a\times(\hat{\Sigma}_o\cup\{\hat{\epsilon}\})\rightarrow Q_e$ is the transition function from attack states to environment states defined by: for any $q_a=(q,\tilde{q},z,\sigma)$, we have $\Delta_M(q_a)=\mathcal{V}(\sigma)$. For each $\hat{\sigma}_a\in\Delta_M(q_a)$, we have $f_{ae}(q_a,\hat{\sigma}_a)=(q',\tilde{q}',z')$, where $q'=\mathsf{NX}_{\sigma_a}(\mathsf{UR}_{\Delta_H(z)}(q))$, $\tilde{q}'=\mathsf{NX}_{\sigma}(\mathsf{UR}_{\Delta_H(z)}(\tilde{q}))$ 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. The system-player plays at each *environment state* by randomly generating a feasible observation in $\mathcal{O}(\tilde{q}, \Delta_H(z))$. Whenever an observation σ occurs, the game moves to an attack state simply by "remembering" σ. Note that we use a new state z_{att} to denote that the supervisor has detected the presence of the attacker. Therefore, if the third component of the state is z_{att}, the game stops, i.e., no active event is defined. The attacker-player plays at each *attacker states* by choosing a doctored observation. Note that, when the attacker changes the original observation σ to σ_a, we use notation $\hat{\sigma}_a$ with a hat to emphasize that this is a doctored observation. For technical reason, we define $\Delta_H(z_{\mathsf{att}}) = \emptyset$.

To formally connect the AAS structure with the attacker strategies, we introduce the following concepts. First. we denote by $\mathcal{L}_e(M) = \{\alpha \in \Sigma_M^* : f(\alpha) \in Q_e\}$ the set of *extended strings* where all strings end up with an environment state. Consider $\alpha = \sigma_1\hat{\sigma}_{a1}\sigma_2\hat{\sigma}_{a2}\cdots\sigma_n\hat{\sigma}_{an} \in \mathcal{L}_e(M)$. We denote by $\mathsf{obs}(\alpha) = \sigma_1\sigma_2\cdots\sigma_n$ the string consists of odd events in α, and denote by $\mathsf{tam}(\alpha) = \sigma_{a1}\sigma_{a2}\cdots\sigma_{an}$ the string consists of even events in α with "hats" removed. Therefore, for $\alpha \in \mathcal{L}_e(M)$, $\mathsf{obs}(\alpha)$ is the actual observation, which is observed by the attacker, and $\mathsf{tam}(\alpha)$ is the modified observation, which is observed by the supervisor. The inverse obs^{-1} is defined by: for every observation $\alpha = \sigma_1\cdots\sigma_n \in P(\mathcal{L}(G))$, we have $\mathsf{obs}^{-1}(\alpha) = \{\sigma_1\}\mathcal{V}(\sigma_1)\cdots\{\sigma_n\}\mathcal{V}(\sigma_n)$.

On the other hand, given an attack strategy A and an actual observation $\alpha = \sigma_1\sigma_n\ldots\sigma_n \in P(\mathcal{L}(S_A/G))$, an extended string can be uniquely specifies as $\alpha_A = \sigma_1\hat{\sigma}_{a1}\sigma_2\hat{\sigma}_{a2}\cdots\sigma_n\hat{\sigma}_{an}$ where $\sigma_{ai} = A(\sigma_1\sigma_2\cdots\sigma_i)$. Clearly, we have $\mathsf{obs}(\alpha_A) = \alpha$ and $\mathsf{tam}(\alpha_A) = g_A(\alpha)$.

We denote by $Q_{\mathsf{att}} = \{(q, \tilde{q}, z) \in Q_e : z = z_{\mathsf{att}}\}$ the set of *attack-revealing* states. Then the following result says that the attack remains stealthy along a string if and only if it does not reach an attack-revealing state in M.

Lemma 1. For any attacker A and observation $\alpha\sigma \in P(\mathcal{L}(S_A/G))$ such that A is stealthy along α, let $q_e = (q, \tilde{q}, z) = f((\alpha\sigma)_A)$ be the environment state reached by $(\alpha\sigma)_A$. Then A is stealthy along $\alpha\sigma$ if and only if $q_e \notin Q_{\mathsf{att}}$.

Example 5. Still, let us consider system G and supervisor S shown in Figure 1. Figure 4 shows part of the AAS M. The initial state is $(X_1, \tilde{X}_1, z_0)$, which is the tuple of the initial-state sets of G, $\tilde{G}$ and H. From the initial-state, the system randomly chooses an event in $\{b, c\} = \mathcal{O}_{\tilde{G}}(\tilde{X}_1, \Delta_H(z_0))$ to play. If event b is chosen, the AAS will enters attacker state a_{01}. The attacker can choose an event in $\mathcal{V}(b) = \{\hat{\epsilon}, \hat{b}\}$ to play. If $\hat{\epsilon}$ is chosen, the supervisor observes nothing. Therefore, the first and the third components do not change. However, the attacker knows that b is generated and therefore, the second component is updated to $\tilde{X}_2 = \widetilde{\mathsf{NX}}_b(\widetilde{\mathsf{UR}}_{\Delta_H(z_0)}(\tilde{X}_1)) = \{(1,3), (1,4), (2,4)\}$. As a result, AAS enters environment state $(X_1, \tilde{X}_2, z_0)$. Again, the system randomly chooses an event in $\{d, c\} = \mathcal{O}_{\tilde{G}}(\tilde{X}_2, \Delta_H(z_0))$ to play. If event d is chosen, the AAS will enters attacker state a_{03}. Since $\mathcal{V}(d) = \{\hat{d}\}$, the attacker can only choose $\hat{d}$ to play, i.e., it cannot modify the observation. However, in the control logic of supervisor

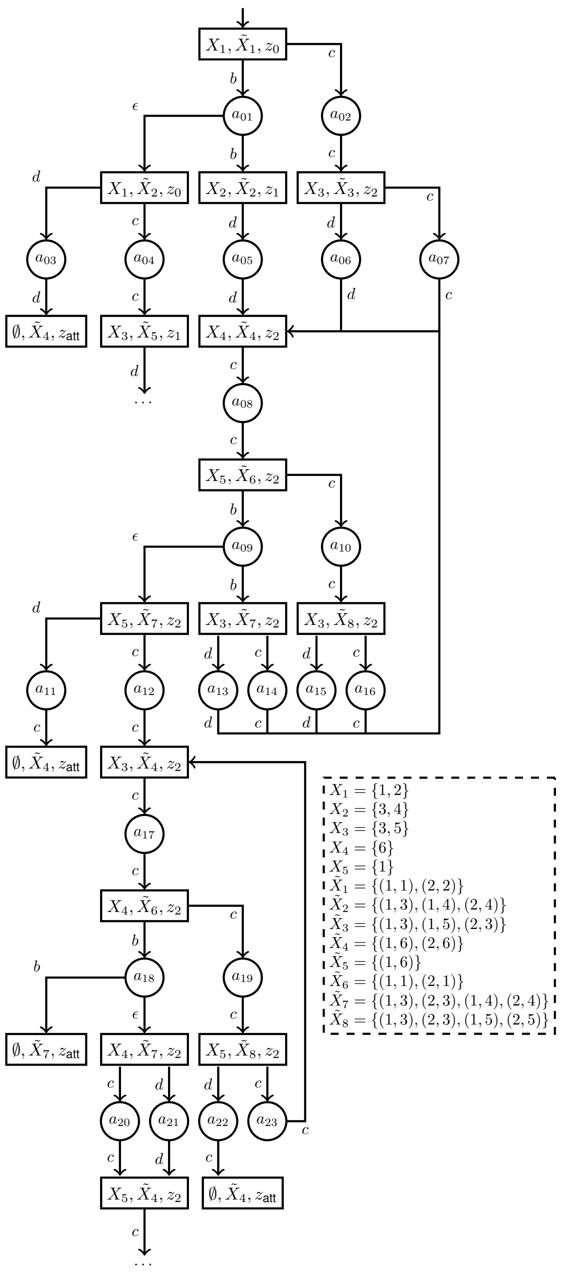

Fig. 4. (Part of) the AAS for system G and supervisor S for Example 5.

S, $d \notin \mathcal{O}(X_1, \Delta_H(z_0))$. Therefore, the supervisor will be aware of presence of the attacker. As a result, AAS M enters a attack-revealing state $(\emptyset, \tilde{X}_4, z_{\mathsf{att}})$. The remaining part of the AAS structure is constructed in the same way.

V. The Simplified AAS

In the previous section, we have introduced the AAS that embeds all attack strategies until their presence is revealed. In practice, once the attacker knows for sure that the system was or was not initiated from a secret state, the attacker-player's win or loss has been completely determined. Based on this idea, the AAS can be simplified.

Definition 3 (Detected States)**.** Given AAS M, for any environment state $q_e = (q, \tilde{q}, z) \in Q_e$, we say q_e is a

- *positive detected state* if $I(\tilde{q}) \subseteq X_{sec}$; and

- *negative detected state* if $I(\tilde{q}) \cap X_{sec} = \emptyset$.

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

Furthermore, 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, this observation is summarized by the following result.

Proposition 1. Let $q_e \in Q_M$ be a positive (respectively, negative) detected state. Then all environment states reachable from q_e are positive (respectively, negative) detected states.

Positive/negative detected environment states essentially provide a state-based winning condition for the attacker-player. In some cases, we can also determine in advance that the attacker *cannot* win the game even before it reveals itself. This is captured by the notion of undetectable states.

Definition 4 (Undetectable States)**.** Given AAS M, for any environment state $q_e = (q, \tilde{q}, z) \in Q_e$, we say q_e is a *undetectable state* if (i) $I(\tilde{q}) \cap X_{sec} \neq \emptyset$; and (ii) for any $(x_0, x) \in \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$ such that $x_0 \in X_{sec}$, there exists $(x'_0, x) \in \widetilde{\mathsf{UR}}_{\Delta_H(z)}(\tilde{q})$ such that $x'_0 \notin X_{sec}$. We denote by $Q_{ud} \subseteq Q_e$ the set of all undetectable states.

Intuitively, if the system reaches an undetectable state, then it means that *whenever* the system was possibly initiated from a secret state $x_0 \in X_{sec}$, it is also possible that the system was initiated from a non-secret state $x_0 \in X_{sec}$ such that the two cases end up with the same current-state x via trajectories having the same observation. Therefore, once the system reaches an undetectable state, we can conclude immediately that the attacker can no longer detect the secret in the future when actually the initial state is secret.

Proposition 2. Let $q_e \in Q_M$ be an undetectable state in AAS M. Then all environment states reachable from q_e are either undetectable or negative detected states.

Based on Propositions 1 and 2, we know that once we reach a detected state in Q_{det}, the attacker knows for sure whether the initial state is secret, and once we reach an undetectable state in Q_{ud}, the attacker will never be able to discover the initial secret when the truth is secret. Therefore, there is no need to further expand the AAS from detected states and undetectable states, which leads to the simplified AAS (SAAS). Specifically, we denote by

$$M^s = (Q^s, \Sigma_M, f^s, q_0),$$

the *simplified AAS*, which is the reachable part of the AAS M by removing all outgoing transitions from states $Q_{det} \cup Q_{ud}$.

Example 6. For the AAS shown in Example 5, its SAAS is given in Figure 5. For environment state $(X_3, \tilde{X}_5, z_1)$, since $I(\tilde{X}_5) = I(\{(1,6)\}) = \{1\} \subseteq X_{sec}$, it is positive detected. For environment state $(X_4, \tilde{X}_4, z_2)$, since $\widetilde{\mathsf{UR}}_{\Delta_H(z_2)}(\tilde{X}_4) = \{(1,6),(2,6)\}$, where $1 \in X_{sec}$ but $2 \notin X_{sec}$, it is undetectable. In fact, the revealing state $(\emptyset, \tilde{X}_4, z_{\mathsf{att}})$ is also

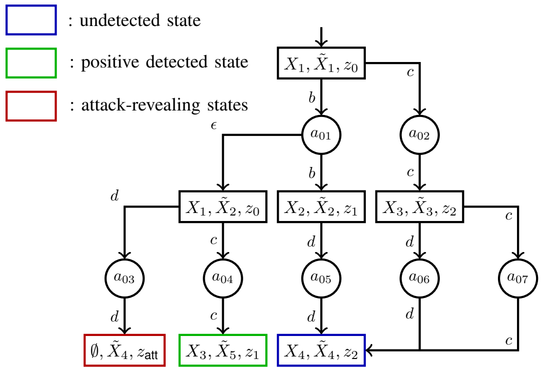

Fig. 5. The simplified AAS of the AAS in Figure 4.

undetectable. Therefore, we can remove active events defined at these states from the AAS, which gives the SAAS a much smaller state space.

VI. Synthesis of Attack Strategies

A. Single Attack Structure

Note that in the SAAS, at each attack state, the attacker may have multiple choices. To synthesize a deterministic attack strategy, the single attacker structure (SAS) is defined, which is a sub-system of the SAAS such that each attack state only has no more than one active event.

Definition 5 (Single Attack Structure)**.** Let $m = (Q^m = Q'_a \cup Q'_e, \Sigma_M, f^m, q_0)$ be a sub-system of the SAAS M^s. We say m is a *single attack structure* if

1) for any state $q_a \in Q'_a$, we have $|\Delta_m(q_a)| = 1$;
2) for any state $q_e \in Q'_e$, we have $|\Delta_m(q_e)| = |\Delta_{M^s}(q_e)|$.

We denote the set of all SAS of the SAAS M^s as $\mathbb{S}(M^s)$.

Intuitively, in a single attack structure, at each attack state, it only has *a unique choice* of attack strategy, and at each environment state, it can reactive to all possible system events. Recall that, for any observation $\alpha \in P(\mathcal{L}(G))$, $\mathsf{obs}^{-1}(\alpha) \in \Sigma^*_M$ is the set of all extended strings with observation α. However, in terms of SAS m, we know that the cardinality of $\mathsf{obs}^{-1}(\alpha) \cap \mathcal{L}_e(m)$ is always smaller than or equal to one. Therefore, we denote by $\mathsf{obs}^{-1}_m(\alpha)$ the unique extended string m such that $\mathsf{obs}(\mathsf{obs}^{-1}_m(\alpha)) = \alpha$ when it exists.

Therefore, given a SAS $m \in \mathbb{S}(M^s)$, we can uniquely an attack strategy A_m by: for any observation $\alpha\sigma \in P(\mathcal{L}(G))$,

- If $\mathsf{obs}^{-1}_m(\alpha)$ exists, then $A_m(\alpha) = \sigma_a$, where $\hat{\sigma}_a$ is the last event of $\mathsf{obs}^{-1}_m(\alpha)$;
- If $\mathsf{obs}^{-1}_m(\alpha)$ does not exist, then $A_m(\alpha) = \sigma$, where σ is the last event of α.

We call such strategy A_m the SAS m *induced strategy*. We can easily show, by induction, that $\alpha_{A_m} = \mathsf{obs}^{-1}_m(\alpha)$ for any observation α. Therefore, by understanding how a SAS can "encode" an attacker strategy, hereafter, we will focus on finding a SAS instead of finding an attacker strategy.

B. Synthesis of Attacker Strategy

Now, we show how to synthesize an attack strategy, which is encoded as a SAS, such that Problem 1 is solved. The following result shows that, in order to ensure an attack

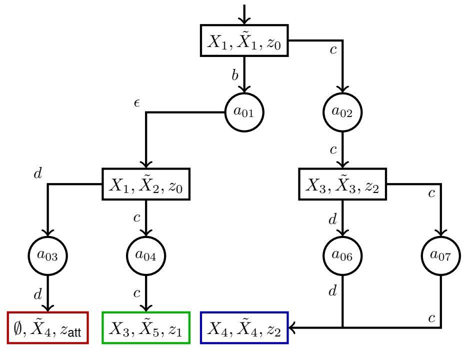

Fig. 6. Synthesized attack strategy represented by a SAS.

sequence in its induce strategy, the necessary and sufficient condition is to have a positive detected state in the SAS.

Theorem 1. Let $m \in \mathbb{S}(M^s)$ be a SAS. Then A_m solves Problem 1, i.e., A_m is IS-detectable, iff $Q^m \cap Q^+_{det} \neq \emptyset$.

The above result says that there exists a SAS-encoded attacker strategy if and only if the SAAS contains a positive detected state. This implicitly restricts our solution space to SAS-encoded attackers. The following result further says that, in fact, such a restriction is without loss of generality for the solvability of Problem 1.

Theorem 2. Supervisory control system S/G is IS-detectable, i.e., Problem 1 has a solution, if and only if there exists a SAS $m \in \mathbb{S}(M^s)$ such that A_m is IS-detectable.

The above two theorems suggest immediately an approach for synthesizing an IS-detectable attack strategy:

- First, we build the SAAS M^s and check whether or not it contains a positive detected state;
- If so, then we find a SAS $m \in \mathbb{S}(M^s)$ such that a positive detected state is contained, and its induced strategy A_m is the solution.

We illustrate the procedure by the following example.

Example 7. Still, in our running example, the SAAS M^s has been shown in Figure 5, where $Q^+_{det} = \{(X_3, X_5, z_1)\} \neq \emptyset$. A possible SAS $m \in \mathbb{S}(M^s)$ is given in Figure 6. Compared with M^s in Figure 5, m in Figure 6 only has one out-going transition at attack state a_{01}. Then according to Theorem 1, A_m is an IS-detectable strategy.

VII. Conclusion

This paper investigated the problem of synthesizing active-sensor attackers against the initial-state opacity of supervisory control systems. To this end, we defined an information structure, call the AAS, that embeds all possible attack strategies. Using the structural properties of initial-state estimation, we further simplified the AAS and based on which, an attacker strategy is synthesized. Note that, in this work, we only require that the synthesized attacker can *possibly* detected the initial secret along some path. This is motivated by the negation of opacity, i.e., if the attacker can potentially threaten the system, then the system is not secure. In some cases, we may further require that the synthesized attacker can *always* detected the initial secret along any path. This synthesis problem with the stronger requirement is currently under investigation. Our preliminary result shows that memory is needed in order to realize such attackers.

References

[1] L.K. Carvalho, Y.-C. Wu, R. Kwong, and S. Lafortune. Detection and mitigation of classes of attacks in supervisory control systems. *Automatica*, 97:121–133, 2018.

[2] J.C. Basilio, C.N. Hadjicostis, and R. Su. 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, 2021.

[3] Z. Ma and K. Cai. On resilient supervisory control against indefinite actuator attacks in discrete-event systems. *IEEE Control Systems Letters*, 2022.

[4] Z. Ma and K. Cai. Optimal secret protections in discrete-event systems. *IEEE Transactions on Automatic Control*, 2022.

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

[6] X. Yin, M. Zamani, and S. Liu. On approximate opacity of cyber-physical systems. *IEEE Transactions on Automatic Control*, 66(4):1630–1645, 2021.

[7] X. Yu, X. Yin, S. Li, and Z. Li. Security-preserving multi-agent coordination for complex temporal logic tasks. *Control Engineering Practice*, 123:105130, 2022.

[8] Y. Ji, X. Yin, and S. Lafortune. Enforcing opacity by insertion functions under multiple energy constraints. *Automatica*, 108:108476, 2019.

[9] S. Liu, A. Trivedi, X. Yin, and M. Zamani. Secure-by-construction synthesis of cyber-physical systems. *Annual Reviews in Control*, 2022.

[10] A. Wintenberg, M. Blischke, S. Lafortune, and N. Ozay. A general language-based framework for specifying and verifying notions of opacity. *Discrete Event Dynamic Systems*, pages 1–37, 2022.

[11] S. Yang, X. Yin, S. Li, and M. Zamani. Secure-by-construction optimal path planning for linear temporal logic tasks. In *59th IEEE Conference on Decision and Control (CDC)*, pages 4460–4466. IEEE, 2020.

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

[13] J. Dubreil, P. Darondeau, and H. Marchand. Supervisory control for opacity. *IEEE Trans. Automatic Control*, 55(5):1089–1100, 2010.

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

[15] X. Yin and S. Lafortune. A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Trans. Automatic Control*, 61:2140–2154, 08 2016.

[16] Y. Tong, Z. Li, C. Seatzu, and A. Giua. Current-state opacity enforcement in discrete event systems under incomparable observations. *Discrete Event Dynamic Systems*, 28(2):161–182, 2018.

[17] Y. Xie, X. Yin, and S. Li. Opacity enforcing supervisory control using non-deterministic supervisors. *IEEE Transactions on Automatic Control*, 2021.

[18] L. Lin and R. Su. Synthesis of covert actuator and sensor attackers. *Automatica*, 130:109714, 2021.

[19] Y. Wang, Y. Li, Z. Yu, N. Wu, and Z. Li. Supervisory control of discrete-event systems under external attacks. *Information Sciences*, 562:398–413, 2021.

[20] M. Alves, P.N. Pena, and K. Rudie. Discrete-event systems subject to unknown sensor attacks. *Discrete Event Dynamic Systems*, pages 1–16, 2021.

[21] R. Su. Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations. *Automatica*, 94:35–44, 2018.

[22] R. Meira-Góes, E. Kang, R.H. Kwong, and S. Lafortune. Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems. *Automatica*, 121:109172, 2020.

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

[24] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*, volume 3. Springer, 2021.

[25] S. Shu and F. Lin. I-detectability of discrete-event systems. *IEEE TASE*, 10(1):187–196, 2012.