Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-28 (2022) 38–45

Sensor and Actuator Attacks in Discrete Event Systems ⋆

Qi Zhang * **Carla Seatzu** ** **Zhiwu Li** * **Alessandro Giua** **

* *SEME, Xidian University, Xi'an 710071, China (e-mail: qzhang_3@stu.xidian.edu.cn, zhwli@xidian.edu.cn)*

** *DIEE, University of Cagliari, 09123 Cagliari, Italy (e-mail: {seatzu, giua}@unica.it)*

Abstract: In supervised discrete event systems under attack, the goal of the supervisor, who has a partial observation of the system evolution, is to prevent the system from reaching a set of unsafe states. An attacker may act in two different ways: it can corrupt the observation of the supervisor by editing the sensor readings, and it can enable events that have been disabled by the supervisor. This is done with the aim of leading the plant to an unsafe state. A special automaton, called attack structure is constructed as the parallel composition of two special structures: an attacker observer and a supervisor under attack. Such an automaton can be used by the attacker to select proper actions (if any) to lead the plant to reach the unsafe state.

Keywords: Discrete event system, sensor attack, actuator attack, supervisory control.

1. INTRODUCTION

Cyber-physical systems (CPS) operating in a feedback loop are particularly vulnerable to attacks since the communication between controllers and processes typically occurs via a network such as the internet. Controllers collect the data from the processes through sensors and based on such data provide a suitable control input (Harirchi and Ozay, 2018). Malicious attackers may corrupt the sensor readings collected by the controller, and/or may alter the control commands (Clark and Zonouz, 2019).

Consider a plant G modeled by a partially-observed discrete event system. The open-loop behaviour of the plant, i.e., the language $L(G)$, may contain undesirable words that should be prevented by a feedback controller. Thus a *partial-observation supervisor* S_P is introduced to restrict $L(G)$ within a sublanguage $K \subseteq L(G)$ by appropriately disabling events in the plant so that the specification is achieved. As a special case of the specification, one may also consider the problem of avoiding a set of *unsafe states* $X_{us} \subseteq X$, where X is the state space of G. In this paper, we deal with the problem of cyber attacks in supervisory control systems. An attacker that is assumed to have a full knowledge of the closed-loop system S_P/G may act in two different ways: *sensor attack* and *actuator attack*. In the first case the attacker may corrupt the sensor channels transmitting erroneous observations to the supervisor. In particular, the attacker may erase the output symbols produced by certain events, and may insert observations corresponding to events that have not occurred. In the second case the attacker corrupts the control commands of the supervisor enabling events that were disabled by the supervisor. Both sensor attack and actuator attack may lead the plant to an unsafe state, thus damages to the system may occur.

The problem of attack detection in CPS is attracting an increasing attention in the automatic control community. It has been investigated in (Pasqualetti et al., 2013; Fawzi et al., 2014) in the case of time-driven continuous systems. Interesting contributions have also been proposed in the discrete event systems framework. As an example, in (Thorsley and Teneketzis, 2006) the authors determine the condition under which the supervisor can detect the presence of an attacker and prevent the system from generating illegal words. They use a language measure technique to assess the damage caused by the attacker if the supervisor cannot block the intrusion, and determine an optimal specification for the supervisor to realize in the presence of an attacker.

Examples of sensor attacks in discrete event systems include (Su, 2018; Wakaiki et al., 2019; Meira-Góes et al., 2020; Meira-Góes and Lafortune, 2020; You et al., 2022). In (Su, 2018), the author defines a robust supervisor against bounded sensor attacks w.r.t. a set of protected observable events that cannot be corrupted by an attacker. Meira-Góes and Lafortune (2020) assume that a plant is controlled by two supervisors; they develop a switching mechanism to determine which supervisor is active at a certain time so that different specifications (e.g. safety, liveness, and maximally permissiveness) could be enforced. Finally, You et al. (2022) propose control policies to enforce safety to the plant under sensor replacement attacks in the context of Petri nets.

Examples of actuator attacks include (Lin et al., 2020; Zhu et al., 2019; Yao et al., 2020; Li et al., 2020). Lin

⋆ This work is partially supported by the National Key R&D Program of China under Grant 2018YFB1700104, the Natural Science Foundation of China under Grand No. 61873342, the ShaanXi Huashan Scholars, the Science and Technology Development Fund, MSAR, under Grant No. 122/2017/A3. This work is also partially supported by Project RASSR05871 MOSIMA funded by Region Sardinia, FSC 2014-2020, Annualita' 2017, Area 3, Action Line 3.1.

10.1016/j.ifacol.2022.10.321

et al. (2020) formulate the actuator attacker synthesis problem as the Ramadge-Wonham supervisor synthesis problem. The authors of (Zhu et al., 2019) consider actuator enablement attacks, and develop a behavior-preserving supervisor that is robust against such attacks. Yao et al. (2020) transform the attack mitigation problem into a tolerant control problem. Finally, in (Li et al., 2020) the authors assume that actuator attacks and control delays may occur simultaneously.

The simultaneous occurrence of the two kinds of attacks has been considered in (Carvalho et al., 2018; Lima et al., 2019; Lin and Su, 2021; Wang and Pajic, 2019). In (Carvalho et al., 2018; Lima et al., 2019), the authors try to use proper techniques to detect attacks, and develop the security module against such attacks. Lin and Su (2021) address the problem of stealthy sensor and actuator attacker synthesis, i.e., the attacker should not be detected before damages are caused to the system. Finally, the authors of (Wang and Pajic, 2019) generalize the problem of attack-resilient supervisory control by modeling the attacker and the supervisor both using *finite state transducers*. They also consider *replay attack* that has not been studied much in the DES framework.

In this paper we propose an alternative approach for solving the problem of supervisory control under attack, that follows from the notion of *joint estimator* (Zhang et al., 2021) that we have used for solving the problem of state estimation under attack. In more detail, we derive an attack policy associated with both sensor and actuator attacks, based on the notion of *attack structure* that simultaneously keep into account the set of states that are consistent with the real observation generated by the plant and the state of the supervisor based on the corrupted observation. Such an attack structure is computed as the *parallel composition* of two particular structures, called *attacker observer* and *supervisor under attack*. The attack structure can be used to verify the effectiveness of an attack, which is defined via an appropriate *attack function*.

The remainder of the paper is organized as follows. In Section 2, some necessary preliminaries on finite state automata and supervisory control theory are introduced. In Section 3, the attack model is given. In Section 4, the problem statement is presented. In Section 5, the attacker observer and the supervisor under attack are introduced. In Section 6, we introduce the attack structure, which provides the basic tool for solving the problem formalized in Section 4. Indeed, it allows to select (potentially) successful attacks. In Section 7, conclusions are finally drawn and our future lines of research in this framework are pointed out.

2. PRELIMINARIES

Let E be an *alphabet*, we denote as E^* the set of all words on the alphabet. Let $\sigma_1, \sigma_2 \in E^*$ be two words, we denote as $\sigma_1\sigma_2$ their *concatenation*.

A *deterministic finite-state automaton* (DFA) is a 4-tuple $G = (X, E, \delta, x_0)$, where X is a finite set of states, E is an alphabet of events, δ: $X \times E \to X$ is a transition function, and x_0 is an initial state. The transition function determines the dynamics of the DFA: if $\delta(x, e) = x'$, the occurrence of event e at state x yields state x'. The transition function can be extended to δ^*: $X \times E^* \to X$, and we write $\delta^*(x, \sigma) = x'$ to denote that the occurrence of a sequence of events $\sigma \in E^*$ at state x yields state x'. The *language generated by* G is defined as $L(G) = \{\sigma \in E^* \mid (\exists x \in X)\ \delta^*(x_0, \sigma) = x\}$. We denote by $\Gamma_G(x) = \{e \in E \mid (\exists x' \in X)\ \delta(x, e) = x'\}$ the set of events that are *active* at state x of G.

Given two automata $G_1 = (X_1, E_1, \delta_1, x_{01})$ and $G_2 = (X_2, E_2, \delta_2, x_{02})$, the *parallel composition* of G_1 and G_2 is denoted as $G = G_1 \parallel G_2 = (X_1 \times X_2, E_1 \cup E_2, \delta, (x_{01} \times x_{02}))$, where the transition function δ is defined as follows:

$$\begin{cases} \delta[(x_1, x_2), e] = (x_1', x_2') \text{ if } \delta_1(x_1, e) = x_1' \wedge \delta_2(x_2, e) = x_2', \\ \delta[(x_1, x_2), e] = (x_1', x_2) \text{ if } \delta_1(x_1, e) = x_1' \wedge e \notin E_2, \\ \delta[(x_1, x_2), e] = (x_1, x_2') \text{ if } \delta_2(x_2, e) = x_2' \wedge e \notin E_1, \\ \text{undefined} \qquad\qquad \text{otherwise.} \end{cases} \tag{1}$$

We point out that in the parallel composition, if there exist states that are not reachable from the initial state, then such states should be removed.

Consider a plant modeled by a DFA $G = (X, E, \delta, x_0)$, let $E = E_o \cup E_{uo} = E_c \cup E_{uc}$, where E_o is the set of *observable events*, E_{uo} is the set of *unobservable events* (due to sensor limitations), E_c is the set of *controllable events*, and E_{uc} is the set of *uncontrollable events* (due to actuator limitations).

Given two alphabets $E' \subseteq E$, the *natural projection* on E', $P_{E'} : E^* \to (E')^*$ is defined as (Ramadge and Wonham, 1989):

$$P_{E'}(\varepsilon) := \varepsilon,\ P_{E'}(\sigma e) := \begin{cases} P_{E'}(\sigma)e & \text{if } e \in E', \\ P_{E'}(\sigma) & \text{if } e \in E \backslash E'. \end{cases} \tag{2}$$

In simple words, given a word $\sigma \in E^*$, its natural projection on E' is obtained by simply removing events that do not belong to E'. For simplicity, we use $P_o : E^* \to E_o^*$ to denote the natural projection on E_o. The *inverse projection of* P_o denoted by $P_o^{-1} : E_o^* \to 2^{E^*}$ is defined as $P_o^{-1}(s) = \{\sigma \in E^* \mid P_o(\sigma) = s\}$, where $s \in E_o^*$.

The *unobservable reach* of state x is defined by a set of states $x' \in X$ reached from state $x \in X$ by executing an unobservable word $\sigma \in E_{uo}^*$, namely, $UR(x) = \{x' \in X \mid (\exists \sigma \in E_{uo}^*)\ \delta^*(x, \sigma) = x'\}$.

Given a plant $G = (X, E, \delta, x_0)$ with set of observable events E_o, the *observer* of G (Cassandras and Lafortune, 2021) is the DFA $Obs(G) = (B, E_o, \delta_{obs}, b_0)$, where:

- $B \subseteq 2^X$ is the set of states,
- $\delta_{obs} : B \times E_o \to B$ is the transition function defined as:
$$\delta_{obs}(b, e_o) := \bigcup_{x \in b} UR(\{x' \mid \delta(x, e_o) = x'\}),$$
- $b_0 := UR(x_0)$ is the initial state.

In this paper we assume that the control action of the supervisor is described by means of a DFA $S_P = (Y, E_o, \delta_s, y_0)$.

We consider the following two assumptions:

(A1) $E_c \subseteq E_o$,
(A2) $L(S_P) \subseteq P_o[L(G)]$.

The first assumption implies that unobservable events are also uncontrollable, thus they could never be disabled by the supervisor. The fact that the supervisor is defined on the alphabet E_o guarantees the admissibility of the control. We notice that Assumption (A1) is not necessary for our approach but allows to simplify the presentation and will be removed in our future work.

The second assumption is done for sake of simplicity but can be easily removed by making the parallel composition of an arbitrary supervisor and the observer of the system.

The supervisor S_P computes a *control function* $f_c : P_o[L(G)] \to 2^E$. Given a word $\sigma \in E^*$ generated by the system, the corresponding *control pattern* is equal to $\xi = f_c[P_o(\sigma)]$ and coincides with the set of events enabled by the supervisor at state $y = \delta_s^*[y_0, P_o(\sigma)]$, namely, it is $\xi = \Gamma_{S_P}(y)$. The resulting closed-loop system is an automaton denoted as $S_P/G = S_P \parallel G$, with generated language $L(S_P/G)$.

3. CLOSED-LOOP SYSTEM UNDER ATTACK

In this chapter we consider a closed-loop system S_P/G subject to attacks according to the scheme in Fig. 1. If σ is a generic word generated by the plant, the *observed word* is $s = P_o(\sigma)$. An attacker may corrupt the observation (sensor attack), inserting fake observations or erasing some output signals produced by events that have actually happened. Such a *corrupted observation* is denoted by s' and is still a sequence of events in E_o. The supervisor constructs its *state estimation* based on s' and elaborates its control pattern based on it. In addition, the attacker may enable some events that were disabled by S_P (actuator attack). We denote as $\xi \in 2^E$ the *control pattern* computed based on s', and denote as $\xi' \in 2^E$ the *corrupted control pattern* that actually restricts the behavior of G.

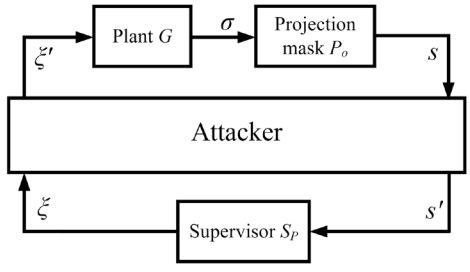


Fig. 1. Closed-loop system under attack.

The set of *attackable events*, namely the events that can be corrupted by the attacker, is denoted by E_{att}. Set E_{att} includes: events that the attacker may insert in the supervisor observation even if they have not actually occurred, events that the attacker may erase in the supervisor observation, and events that the attacker may enable even if they were disabled by the supervisor. The above three sets are denoted respectively, E_{ins}, E_{era}, and E_{ena}. In particular, it is E_{ins}, $E_{era} \subseteq E_o$, and $E_{ena} \subseteq E_c$. We assume that such sets are not necessarily disjoint.

We point out that the notion of attackable events was first proposed by Meira-Góes et al. (2020). However, they consider sensor attacks, thus events in (Meira-Góes et al., 2020) may only be of the first two types, i.e., E_{ins} and E_{era}. In this paper, we slightly generalize such a definition also dealing with actuator attacks. As a result, events in E_{att} may also be of the third type above, i.e., $E_{att} = E_{ins} \cup E_{era} \cup E_{ena}$.

The *attack alphabet* (Meira-Góes et al., 2020) is defined as $E_a = E_o \cup E_+ \cup E_-$, and we assume that E_o, E_+, and E_- are disjoint sets. The set of *inserted events* is denoted as E_+, namely $E_+ = \{e_+ \mid e \in E_{ins}\}$. The occurrence of $e_+ \in E_+$ implies that the attacker inserts in the supervisor observation an event e that has not actually happened. The set of *erased events* is denoted by E_-, namely $E_- = \{e_- \mid e \in E_{era}\}$. The occurrence of $e_- \in E_-$ means that the attacker erases from the supervisor observation event e occurred in the plant. In the following, we use $w \in E_a^*$ to denote a word on attack alphabet E_a, and call it an *attack word*.

In addition we assume that the following relationship holds: $E_c \subseteq E_o$, namely all the events that are controllable are also observable. This implies that $E_{uo} \subseteq E_{uc}$, i.e., all the unobservable events are enabled by the supervisor, which simplifies our problem.

The following definition formalizes the notion of attacker via the sensor and actuator attack functions.

***Definition** 1.* Consider a closed-loop system S_P/G where $G = (X, E, \delta, x_0)$, the set of attackable events is $E_{att} = E_{ins} \cup E_{era} \cup E_{ena}$, and the set of observable events is E_o. An attacker is defined by a *sensor attack function* $f_{sen} : P_o(L(S_P/G)) \to E_a^*$, and an *actuator attack function* $f_{act} : P_o[L(S_P/G)] \to 2^E$, where $E_a = E_o \cup E_+ \cup E_-$ is the attack alphabet, and 2^E is the set of control patterns. The attack functions satisfy the following conditions:

(a) $f_{sen}(\varepsilon) \in E_+^*$,
(b) $\forall se \in P_o(L(S_P/G)) \subseteq E_o^*$:

$$\begin{cases} f_{sen}(se) \in f_{sen}(s)\{e_-, e\}E_+^* & \text{if} \quad e \in E_{era}, \\ f_{sen}(se) \in f_{sen}(s)eE_+^* & \text{if} \quad e \in E_o \setminus E_{era}. \end{cases} \tag{3}$$

(c) $\forall s \in P_o[L(S_P/G)]$:

$$f_{act}(s) \subseteq \xi \cup E_{ena} \tag{4}$$

where $\xi = f_c(s')$ is the control pattern computed by the supervisor based on the corrupted word s' corresponding to observation s. ◇

In Definition 1, condition (a) means that the attacker can insert any word in E_+^* when no observable event has occurred in the plant. Condition (b) indicates that if an event $e \in E_{era}$ happens, the attacker can either erase event e or not erase it, and then insert any word in E_+^*. If an event $e \in E_o \setminus E_{era}$ happens, the attacker can insert any word in E_+^* after e. Condition (c) implies that the attacker can enable events in E_{ena} that were disabled by the supervisor.

Note that, based on Definition 1, the attacker can insert an arbitrarily long sequence of events in E_+ between the occurrence of two consecutive observable events of the plant. However, since this could be unrealistic in practice, one can use the *n-bounded attack automaton* G_n proposed in (Zhang et al., 2021) to impose that the attacker can only insert n fake events between two consecutive observable events.

***Definition** 2.* Given a closed-loop system S_P/G under attack, the *attacker mask* $\widetilde{P} : E_a^* \to E_o^*$ is defined as:

$$\widetilde{P}(\varepsilon) = \varepsilon,\ \widetilde{P}(we') = \begin{cases} \widetilde{P}(w)e \text{ if } e' = e \in E_o \vee \\ \qquad\qquad e' = e_- \in E_-, \\ \widetilde{P}(w) \text{ if } e' = e_+ \in E_+. \end{cases} \tag{5}$$

◇

In plain words, the attacker knows that $e_- \in E_-$ are events that have been erased; thus it treats them in the same way in case of event $e \in E_o$. Since the attacker knows that $e_+ \in E_+$ are fake events that have not actually occurred in the plant, it treats events $e_+ \in E_+$ as no events happen.

***Definition** 3.* The *supervisor mask* $\widehat{P} : E_a^* \to E_o^*$ is defined as:

$$\widehat{P}(\varepsilon) = \varepsilon,\ \widehat{P}(we') = \begin{cases} \widehat{P}(w)e \text{ if } e' = e \in E_o \vee \\ \qquad\qquad e' = e_+ \in E_+, \\ \widehat{P}(w) \text{ if } e' = e_- \in E_-. \end{cases} \tag{6}$$

◇

In simple words, the supervisor mask describes how the supervisor deals with events in E_a, i.e., the supervisor cannot distinguish event $e_+ \in E_+$ from the corresponding event $e \in E_o$, and it cannot observe event $e_- \in E_-$. Note that the corrupted observation s' in Fig. 1 and eq. (4) can be written in terms of the supervisor mask $\widehat{P}$, namely it is $s' = \widehat{P}[f_{sen}(s)]$.

Since the attacker can corrupt the observation and the control commands of the supervisor, it may happen that the supervisor observes a word $s \notin P_o[L(S_P/G)]$, this means that the supervisor detects the presence of an attacker. In the following, we assume that the supervisor disables all the controllable events once it discovers the presence of the attacker. This is formalized modifying the control function f_c as follows.

***Definition** 4.* Consider a closed-loop system S_P/G. Let $\sigma \in L(G)$ be a word generated by G, $s = P_o(\sigma)$, and f_c be the control function of S_P. The *extended control function* $\widehat{f}_c : P_o[L(G)] \to E_o^*$ is defined as:

$$\widehat{f}_c(s) = \begin{cases} f_c(s) \text{ if } s \in P_o[L(S_P/G)] \\ E_{uc} \text{ if } s \notin P_o[L(S_P/G)]. \end{cases} \tag{7}$$

◇

***Definition** 5.* Consider a closed-loop system S_P/G. Let f_{sen} be the sensor attack function, f_{act} be the actuator attack function, $\widetilde{P}$ be the attacker mask, and $\widehat{P}$ be the supervisor mask. The language modified by the attacker is called *language under attack* $L_a(S_P/G) \subseteq E_a^*$, and is recursively defined as follows:

(1) $f_{sen}(\varepsilon) \in L_a(S_P/G)$,
(2) $w \in L_a(S_P/G)$, $e \in (f_{act}[\widetilde{P}(w)] \cup E_{uc}) \cap E_o$, $\widetilde{P}(we) \in P_o[L(G)] \Longrightarrow wf_{sen}(e) \in L_a(S_P/G)$. ◇

In Definition 5, item (1) provides the initial condition of the language under attack. Item (2) means that when word $w \in E_a^*$ has been produced, the plant G receives the corrupted control pattern computed by function f_{act}. Thus a new observable event e may occur only if either it is enabled by the corrupted control pattern or it is uncontrollable, and furthermore the plant, which has previously produced observation $\widetilde{P}(w)$, should be able to generate event e. When such event e occurs after w, the attacker produces the new attack word $wf_{sen}(e)$.

The internal structure of the attacker, which in Fig. 1 is represented as a black box having the observation s and the control pattern ξ as inputs, and the corrupted observation s' and the corrupted control pattern ξ' as outputs, is depicted in Fig. 2.

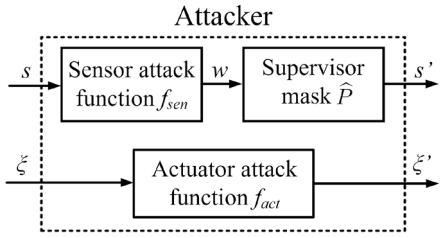


Fig. 2. Internal structure of the attacker in Fig. 1.

Fig. 2 shows how the observation s is corrupted by the sensor attack function f_{sen}, producing the attack word $w \in E_a^*$. Such a sequence is projected via the supervisor mask $\widehat{P}$ on E_o, generating a word s'. The supervisor constructs its state estimation based on s' and computes its control pattern ξ. Such a control pattern is altered by the actuator attack function f_{act}, producing the corrupted control pattern ξ', which actually restricts the behavior of the plant.

4. PROBLEM FORMULATION

We assume that a set of unsafe states X_{us} is given, which corresponds to an undesirable or dangerous condition for the plant G. The supervisor controls the plant with the objective of preventing it from reaching the unsafe state. The goal of the attacker is that of preventing the supervisor from reaching its objective. The following definition provides a criterion to assess the attacker.

***Definition** 6.* Consider a closed-loop system S_P/G with set of attackable events E_{att}. Let X_{us} be a set of unsafe states of plant G, $L_a(S_P/G)$ be the language under attack, and $Obs(G) = (B, E_o, \delta_{obs}, b_0)$ be the observer. An attack function (either a sensor attack function or an actuator attack function) is:

- *successful* if $\exists w \in L_a(S_P/G)$: $\delta_{obs}[b_0, \widetilde{P}(w)] \subseteq X_{us}$;
- *potentially successful* if $\exists w \in L_a(S_P/G)$: $\delta_{obs}[b_0, \widetilde{P}(w)] \cap X_{us} \neq \emptyset$ and $\delta_{obs}[b_0, \widetilde{P}(w)] \not\subseteq X_{us}$. ◇

In simple words, an attack function is successful if there exists at least one attack word $w \in L_a(S_P/G)$ such that executing $\widetilde{P}(w)$ starting from the initial state b_0, the observer reaches a state that is included in the set of unsafe states X_{us}. This means that the plant reaches an unsafe state when the attack word w is generated.

An attack function is potentially successful if there exists at least one attack word $w \in L_a(S_P/G)$ such that executing $\widetilde{P}(w)$ starting from b_0, the observer reaches a state that contains an element in X_{us}, and it is not included in X_{us}. It implies that there is some possibility that the

plant reaches an unsafe state when the attack word w is generated.

In this paper, given a closed-loop system S_P/G with set of attackable events E_{att}, we want to provide a criterion to establish if an attack strategy exists, which leads the plant to an unsafe state.

5. ATTACKER OBSERVER AND SUPERVISOR UNDER ATTACK

In this section we first recall the notion of attacker observer defined in (Zhang et al., 2021), then introduce the definition of supervisor under attack, which provide the basis for the solution of the problem formulated in the previous section.

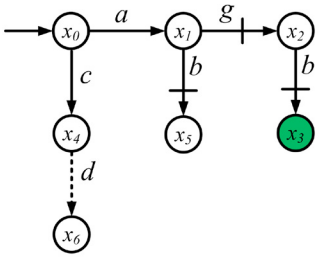
5.1 Attacker Observer

The attacker observer describes all words on alphabet E_a obtained by attacking the observations and the corresponding state estimation of the attacker, which can be constructed using Algorithm 1 in (Zhang et al., 2021).

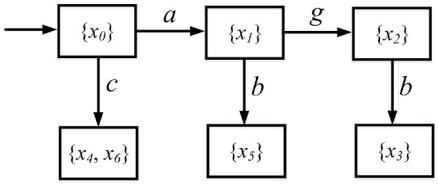
Consider a plant $G = (X, E, \delta, x_0)$ with set of attackable events E_{att}. First, the observer $Obs(G) = (B, E_o, \delta_{obs}, b_0)$ of the plant is constructed. The attacker observer is denoted as $Obs_{att}(G) = (B, E_a, \delta_{att}, b_0)$. The set of states of $Obs_{att}(G)$ coincides with the set of states of $Obs(G)$, as well as the initial state. The attack alphabet $E_a = E_o \cup E_+ \cup E_-$. The transition function is initialized at $\delta_{att} := \delta_{obs}$.

Then, for all events $e \in E_{ins}$ and for all states $b \in B$, we add self-loops $\delta_{att}(b, e_+) = b$. Finally, for all events $e \in E_{era}$ and for all states $b \in B$, whenever a transition $\delta_{att}(b, e)$ is defined, we impose $\delta_{att}(b, e_-) = \delta_{att}(b, e)$.

***Example** 1.* Consider the plant $G = (X, E, \delta, x_0)$ and the observer in Figs. 3(a) and (b), respectively. Let $E_o = \{a, b, c, g\}$, $E_{uo} = \{d\}$, $E_c = \{b, g\}$, $E_{uc} = \{a, c, d\}$, and $X_{us} = \{x_3\}$.



(a) G



(b) $Obs(G)$

Fig. 3. (a) Plant G and (b) Observer $Obs(G)$ in Example 1.

Let $E_{ins} = \{a\}$ and $E_{era} = \{a, g\}$. The attacker observer $Obs_{att}(G)$ is visualized in Fig. 4. Since $a \in E_{ins}$, we add

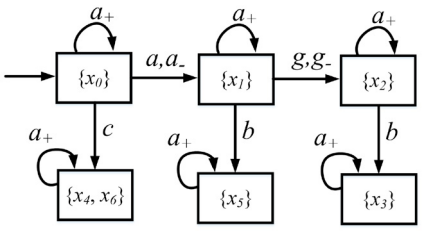


Fig. 4. Attacker observer $Obs_{att}(G)$ in Example 1.

self-loops labeled a_+ at all the states. Since $a, g \in E_{era}$, arcs labeled a and g, respectively, are also labeled a_- and g_-, respectively. ◇

In the following, we denote as $\mathcal{F}_{sen}$ the set of all possible attack functions satisfying eq. (3). The *union of all the attack languages*, denoted as $L(\mathcal{F}_{sen}, G)$, is defined as

$$L(\mathcal{F}_{sen}, G) = \bigcup_{f_{sen} \in \mathcal{F}_{sen}} f_{sen}(P_o[L(G)]).$$

The following proposition provides a characterization of the attacker observer. Its proof can be found in (Zhang et al., 2021).

***Proposition** 1.* Given a plant G with set of attackable events E_{att} and observer $Obs(G) = (B, E_o, \delta_{obs}, b_0)$. Let f_{sen} be a sensor attack function, $\mathcal{F}_{sen}$ be the set of sensor attack functions, and $Obs_{att}(G) = (B, E_a, \delta_{att}, b_0)$ be the attacker observer. The following statements hold:

(a) $L(Obs_{att}(G)) = L(\mathcal{F}_{sen}, G)$;
(b) $\forall s \in P(L(G)), \forall f_{sen} \in \mathcal{F}_{sen}$ with $w = f_{sen}(s) \in E_a^*$:

$$\delta^*_{att}(b_0, w) = \delta^*_{obs}(b_0, s).$$

5.2 Supervisor Under Attack

The supervisor under attack S_{Pa} provides the evolution of the supervisor on the basis of the attack words $w \in E_a^*$. It generates two different sets of words. The first set contains all words on E_a that may result from either an uncorrupted observation or from a corrupted observation that keeps the attacker covert. The second set contains all the words that are not consistent with the uncorrupted observation. While words in the first set lead to a state that according to the supervisor are consistent with the perceived observation, those in the second set lead to a dummy state denoted by $y_\emptyset$. The supervisor under attack S_{Pa} can be computed using Algorithm 1.

We briefly explain how Algorithm 1 works. Consider a supervisor $S_P = (Y, E_o, \delta_s, y_0)$. First, the set of states Y_a is defined as $Y \cup \{y_\emptyset\}$, where $y_\emptyset$ is a dummy state: the supervisor detects the presence of an attacker when $y_\emptyset$ is reached. Moreover, the attack alphabet E_a is computed, and the transition function of S_{Pa} is initialized at δ_s.

Then, for all $e \in [(E_o \cap E_{uc}) \cup E_{ena}]$, and for all $y_a \in Y_a$, if a transition $\delta_{sa}(y_a, e)$ is not defined, then we impose $\delta_{sa}(y_a, e) = y_\emptyset$ (Step 7). Steps 4–10 ensure that the supervisor always enables uncontrollable events. Note that since events in E_{ena} can be enabled by the attacker even if they were disabled by the supervisor, such events are also uncontrollable from the supervisor's viewpoint.

Algorithm 1 Construction of the supervisor under attack S_{Pa}

Input: Supervisor $S_P = (Y, E_o, \delta_s, y_0)$, event sets E_{ins}, E_{era}, and E_{ena}.
Output: Supervisor under attack $S_{Pa} = (Y_a, E_a, \delta_{sa}, y_0)$.

1: Let $Y_a := Y \cup y_\emptyset$;
2: Let $E_a := E_o \cup E_+ \cup E_-$;
3: Let $\delta_{sa} := \delta_s$;
4: **for all** $e \in [(E_o \cap E_{uc}) \cup E_{ena}]$, **do**
5: **for all** $y_a \in Y_a$, **do**
6: **if** $\delta_{sa}(y_a, e)$ is not defined, **then**
7: $\delta_{sa}(y_a, e) = y_\emptyset$;
8: **end if**
9: **end for**
10: **end for**
11: **for all** $e \in E_{ins}$, **do**
12: **for all** $y_a \in Y_a$, **do**
13: **if** $\delta_{sa}(y_a, e)$ is defined, **then**
14: $\delta_{sa}(y_a, e_+) = \delta_{sa}(y_a, e)$;
15: **else**
16: $\delta_{sa}(y_a, e_+) = y_\emptyset$;
17: **end if**
18: **end for**
19: **end for**
20: **for all** $e \in E_{era}$, **do**
21: **for all** $y_a \in Y_a$, **do**
22: **if** $\delta_{sa}(y_a, e)$ is defined, **then**
23: $\delta_{sa}(y_a, e_-) = y_a$;
24: **end if**
25: **end for**
26: **end for**

Finally, for all events $e \in E_{ins}$, and for all states $y_a \in Y_a$, if a transition $\delta_{sa}(y_a, e)$ is defined, then we impose $\delta_{sa}(y_a, e_+) = \delta_{sa}(y_a, e)$ (Step 14). It implies that the supervisor cannot distinguish event e_+ from the corresponding event $e \in E_{ins}$. If a transition $\delta_{sa}(y_a, e)$ is not defined, then we impose $\delta_{sa}(y_a, e_+) = y_\emptyset$ (Step 16). This means that the supervisor observes an event that has been disabled by itself; thus it knows that the plant is under attack. For all events $e \in E_{era}$, and for all states $y_a \in Y_a$, whenever a transition $\delta_{sa}(y_a, e)$ is defined, we add a self-loop $\delta_{sa}(y_a, e_-) = y_a$ (Step 23). In fact, the supervisor cannot observe $e_- \in E_-$.

***Example** 2.* Consider the supervisor in Fig. 5(a). Let $E_o = \{a, b, c, g\}$, $E_{uo} = \{d\}$, $E_c = \{b, g\}$, $E_{uc} = \{a, c, d\}$, $E_{ins} = \{a\}$, $E_{era} = \{a, g\}$, and $E_{ena} = \{b\}$. The supervisor under attack S_{Pa} is visualized in Fig. 5(b).

Since $a, b, c \in [(E_o \cap E_{uc}) \cup E_{ena}]$, and transition labeled b is not defined at state y_0, we add the transition $\delta_{sa}(y_0, b) = y_\emptyset$. Similar arguments can be used to explain the other transitions labeled a, b and c that yield state $y_\emptyset$, including self-loops at state $y_\emptyset$.

Since $a \in E_{ins}$, and there is a transition $\delta_{sa}(y_0, a) = y_1$, we add the transition $\delta_{sa}(y_0, a_+) = y_1$. Similar arguments can be used to explain the other transitions labeled a_+.

Since $a, g \in E_{era}$, and there is a transition $\delta_{sa}(y_0, a) = y_1$, we add a self-loop $\delta_{sa}(y_0, a_-) = y_0$. Similar arguments can be used to explain the other self-loops labeled a_- and g_-. ⋄

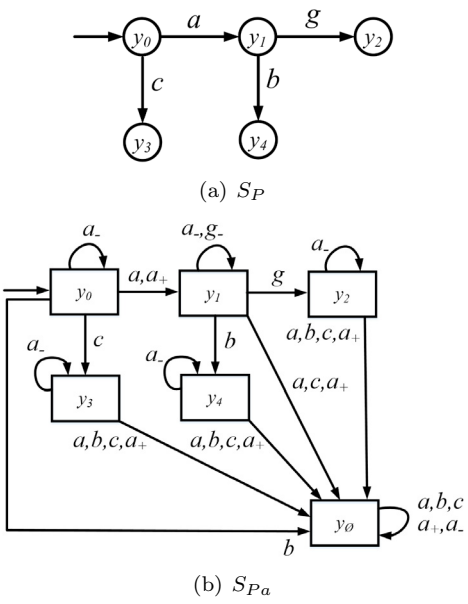


Fig. 5. (a) Supervisor S_P and (b) Supervisor under attack S_{Pa} in Example 2.

In order to characterize the language generated by the supervisor under attack, we first introduce the definition of *stealthy attack function*.

***Definition** 7.* Consider a closed-loop system S_P/G with set of attackable events E_{att}. Let $L_a(S_P/G)$ be the language under attack, and $\widehat{P}$ be the supervisor mask. An attack function is said to be *stealthy* if:

$$\widehat{P}(L_a(S_P/G)) \subseteq L(S_P). \tag{8}$$

⋄

In plain words, an attack function is stealthy if the resulting words can also be observed by the supervisor when no attack occurs.

In the following, the set of *stealthy words* of the supervisor under attacker S_{Pa} is defined as:

$$W_s = \{w \in L(S_{Pa}) \mid \widehat{P}(w) \in L(S_P)\}, \tag{9}$$

while the set of *exposing words* of S_{Pa} is defined as:

$$W_e = L(S_{Pa}) \setminus W_s. \tag{10}$$

Thus W_s is the set of words that are consistent with observations by the supervisor without the presence of an attacker. Hence, the observation of such words does not reveal that the closed-loop system is under attack, while W_e contains words that expose the attacker.

***Proposition** 2.* Consider a closed-loop system S_P/G, where $S_P = (Y, E_o, \delta_s, y_0)$. Let $S_{Pa} = (Y_a, E_a, \delta_{sa}, y_0)$ be the supervisor under attack with a set of stealthy words W_s and a set of exposing words W_e. The following two statements hold:

(a) $L(S_{Pa}) = W_s \cup W_e$;
(b) $\forall w \in L(S_{Pa})$: if $w \in W_s$, then $\delta^*_{sa}(y_0, w) = \delta^*_s[y_0, \widehat{P}(w)]$; if $w \in W_e$, then $\delta^*_{sa}(y_0, w) = y_\emptyset$.

Proof. (a) The result holds by the definition of the set of stealthy words (eq. (9)) and exposing words (eq. (10)).

(b) If $w \in W_s$, we prove it by induction on the length of w. If $w = \varepsilon$, the result holds being $\widehat{P}(w) = \varepsilon$. Consider a word $w \in L(S_{Pa})$ with length greater than one. Let $w = w'e'$. Assume that the result holds for w'. Thus $\delta^*_{sa}(y_0, w') = \delta^*_s[y_0, \widehat{P}(w')]$. It is $\delta^*_{sa}(y_0, w) = \delta_{sa}[\delta^*_{sa}(y_0, w'), e']$ and $\delta^*_s[y_0, \widehat{P}(w)] = \delta_s[\delta^*_s[y_0, \widehat{P}(w')], \widehat{P}(e')]$. By the definition of supervisor mask, $\delta^*_s[y_0, \widehat{P}(w)] = \delta_s(\delta^*_s[y_0, \widehat{P}(w')], e)$ if $e' = e \in E_o \vee e' = e_+ \in E_+$; $\delta^*_s[y_0, \widehat{P}(w)] = \delta_s(\delta^*_s[y_0, \widehat{P}(w')], \varepsilon)$ if $e' = e_- \in E_-$. By Algorithm 1, the transition function starting from a generic state $y_a \in Y_a$ is defined in the same way in case of $e \in E_{ins}$ and $e_+ \in E_+$ (Step 14), and it corresponds to a self-loop in the case of $e_- \in E_-$ (Step 23). Thus, it can be concluded that $\delta^*_{sa}(y_0, w) = \delta^*_s[y_0, \widehat{P}(w)]$.

If $w \in W_e$, by Algorithm 1, all the words w end up in the new state $y_\emptyset$, i.e., $\delta^*_{sa}(y_0, w) = y_\emptyset$. □

6. ATTACK STRUCTURE

In this section, we introduce the notion of attack structure which can be defined as follows.

***Definition** 8.* Consider the closed-loop system S_P/G with a set of attackable events E_{att}. Let $Obs_{att}(G) = (B, E_a, \delta_{att}, b_0)$ and $S_{Pa} = (Y_a, E_a, \delta_{sa}, y_0)$ be the attacker observer and supervisor under attack, respectively. The attack structure A w.r.t. S_P/G and E_{att} is a DFA: $A = Obs_{att}(G) \parallel S_{Pa} = (H, E_a, \delta_a, h_0)$, where:

- the set of states is $H = \{(b, y_a) \mid b \in B, y_a \in Y_a\}$;
- the alphabet is $E_a = E_o \cup E_+ \cup E_-$;
- the transition function: $\delta_a((b, y_a), e) = (\delta_{att}(b, e), \delta_{sa}(y_a, e))$ if $e \in \Gamma_{att}(b) \cap \Gamma_{S_{Pa}}(y_a)$, where $\Gamma_{att}(b)$ (resp., $\Gamma_{S_{Pa}}(y_a)$) is the set of active events at state b (resp., y_a) of $Obs_{att}(G)$ (resp., S_{Pa});
- the initial state is $h_0 = (b_0, y_0)$. ◇

The attack structure describes the state estimation computed by the attacker and the evolution of the supervisor according to its own observation.

Now we discuss the computational complexity of building the attack structure A. Given a plant G with set of states X, and the supervisor S_P with set of states Y. The attacker observer $Obs_{att}(G)$ has at most $2^{|X|}$ states, and the supervisor under attacker S_{Pa} has $|Y|+1$ states. Since $A = Obs_{att}(G) \parallel S_{Pa}$, the complexity of constructing A is $O(2^{|X|} \cdot |Y|)$.

***Theorem** 1.* Consider a closed-loop system S_P/G with set of attackable events E_{att}. Let $Obs(G) = (B, E_o, \delta_{obs}, b_0)$ be the observer of G, $S_P = (Y, E_o, \delta_s, y_0)$ be the supervisor, $L(\mathcal{F}_{sen}, G)$ be the union of all the attack languages, W_s be the set of stealthy words, W_e be the set of exposing words, and $A = (H, E_a, \delta_a, h_0)$ be the attack structure. The following two statements hold:

(a) $L(A) = L(\mathcal{F}_{sen}, G) \cap (W_s \cup W_e)$;
(b) $\forall s \in P(L(G)), \forall f_{sen} \in \mathcal{F}_{sen}$ with $w = f_{sen}(s) \in E_a^*$:
  (i) if $w \in W_s$, then $\delta^*_a(h_0, w) = (b, y_a) \iff \delta^*_{obs}(b_0, s) = b$ and $\delta^*_s[y_0, \widehat{P}(w)] = y_a$, where $y_a \neq y_\emptyset$;
  (ii) if $w \in W_e$, then $\delta^*_a(h_0, w) = (b, y_\emptyset) \iff \delta^*_{obs}(b_0, s) = b$ and $\delta^*_s[y_0, \widehat{P}(w)]$ is not defined.

Proof. (a) By Propositions 1 and 2, it is $L[Obs_{att}(G)] = L(\mathcal{F}_{sen}, G)$, and $L(S_{Pa}) = W_s \cup W_e$. Since A is defined as the parallel composition of $Obs_{att}(G)$ and S_{Pa}, having the same alphabet, $L(A)$ is equal to the intersection of their languages.

(b) Let us first consider the case: $w \in W_s$. (If) Assume that $\delta^*_{obs}(b_0, s) = b$ and $\delta^*_s[y_0, \widehat{P}(w)] = y_a$. By Propositions 1 and 2, it is $\delta^*_{att}(b_0, w) = \delta^*_{obs}(b_0, s)$ and $\delta^*_{sa}(y_0, w) = \delta^*_s[y_0, \widehat{P}(w)]$, i.e., $\delta^*_{att}(b_0, w) = b$ and $\delta^*_{sa}(y_0, w) = y_a$. Since $A = Obs_{att}(G) \parallel S_{Pa}$, by the definition of parallel composition, it holds that $\delta^*_a(h_0, w) = (b, y_a)$.

(Only if) Assume that $\delta^*_a(h_0, w) = (b, y_a)$. Since $A = Obs_{att}(G) \parallel S_{Pa}$, by the definition of parallel composition, it is $\delta^*_{att}(b_0, w) = b$ and $\delta^*_{sa}(y_0, w) = y_a$. By Propositions 1 and 2, it is $\delta^*_{obs}(b_0, s) = \delta^*_{att}(b_0, w)$ and $\delta^*_s[y_0, \widehat{P}(w)] = \delta^*_{sa}(y_0, w)$, i.e., $\delta^*_{obs}(b_0, s) = b$ and $\delta^*_s[y_0, \widehat{P}(w)] = y_a$.

We finally consider the case: $w \in W_e$. The proof follows from the definition of parallel composition and the fact that a word w in the supervisor under attack S_{Pa} leads to state $y_\emptyset$ if and only if $\widehat{P}(w) \notin L(S_P)$, i.e., $\delta_s(y_0, \widehat{P}(w))$ is not defined. □

***Definition** 9.* Let $A = (H, E_a, \delta_a, h_0)$ be an attack structure, and X_{us} be the set of unsafe states of the plant G.

- The set of target states of A is $H_t := \{h = (b, y_a) \in H \mid b \subseteq X_{us}\}$;
- The set of potential target states of A is $H_{pt} := \{h = (b, y_a) \in H \mid b \cap X_{us} \neq \emptyset \wedge b \not\subseteq X_{us}\}$. ◇

The first entry of a target state is included in X_{us}. The closed-loop system reaches an unsafe state when such a state is reached.

The first entry of a potential target state contains an element in X_{us}, but it is not included in X_{us}. The closed-loop system may reach an unsafe state when such a state is reached.

***Theorem** 2.* Consider a closed-loop system S_p/G under attack. Let $Obs(G) = (B, E_o, \delta_{obs}, b_0)$ be the observer of G, $Obs_{att}(G) = (B, E_a, \delta_{att}, b_0)$ be the attacker observer, and $A = (H, E_a, \delta_a, h_0)$ be the attack structure. The following two statements hold:

(a) There exists a successful attack function iff A contains at least one target state.
(b) There exists a potentially successful attack function iff A contains at least one potential target state.

Proof. Let us first consider item (a). (If) Assume that A contains a target state $h = (\{b\}, \{y_a\})$ such that $\delta^*_a(h_0, w) = (b, y_a)$, and $b \subseteq X_{us}$. According to Proposition 1, it is $\delta_{att}(b_0, w) = \delta^*_{obs}(b_0, s)$. Based on Theorem 1, it is $\delta^*_{obs}(b_0, s) = b$. Since $w = f_{sen}(s)$, in accordance with the definition of attacker mask $\widetilde{P}$, it can be concluded that $\widetilde{P}(w) = s$. As a result, $\delta^*_{obs}[b_0, \widetilde{P}(w)] = b \subseteq X_{us}$. By Definition 6, we can conclude that there exists a successful attack function.

(Only if) Assume that there exists a successful attack function, i.e., there exists an attack word w such that $\delta^*_{obs}(b_0, \widetilde{P}(w)) \subseteq X_{us}$. Based on Proposition 1, and the definition of $\widetilde{P}$, it is $\delta_{att}(b_0, w) = \delta^*_{obs}(b_0, s) = \delta^*_{obs}(b_0, \widetilde{P}(w))$.

In accordance with Theorem 1, it is $\delta_a^*(h_0, w) = (b, y_a)$, and $\delta_{obs}^*(b_0, s) = b$. Thus, $\delta_{obs}^*[b_0, \widetilde{P}(w)] = b \subseteq X_{us}$, i.e., state $h = (b, y_a)$ is a target state.

The proof for item (b) is similar to the proof of item (a). □

***Example** 3.* Consider again the closed-loop system S_P/G in Example 1 and Example 2. The attack structure $A = Obs_{att}(G) \parallel S_{Pa} = (H, E_a, \delta_a, h_0)$ is visualized in Fig. 6.

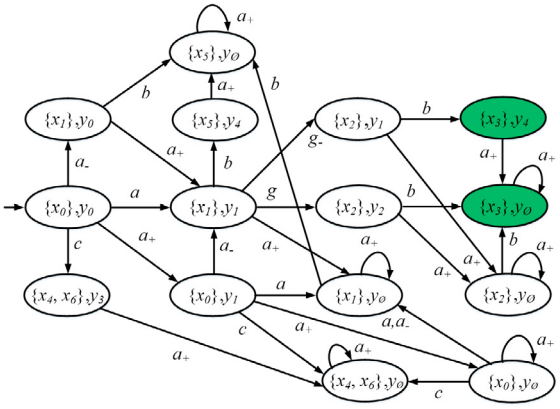


Fig. 6. Attack structure A in Example 3.

In A, two target states are highlighted in green: $(\{x_3\}, y_\emptyset)$ and $(\{x_3\}, y_4)$, thus the attack function is successful. When the target state is reached, the plant is in the unsafe state x_3.

For example, at state $(\{x_1\}, y_1)$, if event $g \in E_{era}$ occurs in the plant, the attacker may erase it, corresponding to the transitions $\delta_a[(\{x_1\}, y_1), g_-] = (\{x_2\}, y_1)$. Then, event b may occur, and the target state $(\{x_3\}, y_4)$ will be reached. ◇

7. CONCLUSIONS AND FUTURE WORK

The problem of cyber attacks has been considered in supervisory control systems. We develop an attack structure computed as the parallel composition of the attacker observer and the supervisor under attack. The attack structure allows the attacker to select attacks that cause the closed-loop system S_P/G to reach an unsafe state. In the future, on the one hand, we plan to synthesize a supervisor that can prevent the plant from reaching the unsafe state even with the presence of an attacker. On the other hand, we will try to solve the problem considered in this paper using Petri nets to see if it can provide a more efficient solution.

REFERENCES

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

Cassandras, C.G. and Lafortune, S. (2021). *Introduction to discrete event systems.* Springer.

Clark, A. and Zonouz, S. (2019). Cyber-physical resilience: Definition and assessment metric. *IEEE Transactions on Smart Grid*, 10(2), 1671–1684.

Fawzi, H., Tabuada, P., and Diggavi, S. (2014). Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Transactions on Automatic Control*, 59(6), 1454–1467.

Harirchi, F. and Ozay, N. (2018). Guaranteed model-based fault detection in cyber-physical systems: A model invalidation approach. *Automatica*, 93, 476–488.

Li, Y., Tong, Y., and Giua, A. (2020). Detection and prevention of cyber-attacks in networked control systems. In *Proceedings of the 15th IFAC Workshop on Discrete Event Systems*, 7–13. Rio de Janeiro, Brazil.

Lima, P.M., Alves, M.V.S., Carvalho, L.K., and Moreira, M.V. (2019). Security against communication network attacks of cyber-physical systems. *Journal of Control, Automation and Electrical Systems*, 30(1), 125–135.

Lin, L. and Su, R. (2021). Synthesis of covert actuator and sensor attackers. *Automatica*, 130, Art. no. 109714.

Lin, L., Zhu, Y., and Su, R. (2020). Synthesis of covert actuator attackers for free. *Discrete Event Dynamic Systems: Theory and Applications*, 30(4), 561–577.

Meira-Góes, R. and Lafortune, S. (2020). Moving target defense based on switched supervisory control: A new technique for mitigating sensor deception attacks. In *Proceedings of the 15th IFAC Workshop on Discrete Event Systems*, 317–323. Rio de Janeiro, Brazil.

Meira-Góes, R., Kang, E., Kwong, R.H., and Lafortune, S. (2020). Synthesis of sensor deception attacks at the supervisory layer of Cyber-Physical Systems. *Automatica*, 121, Art. no. 109172.

Pasqualetti, F., Dörfler, F., and Bullo, F. (2013). Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, 58(11), 2715–2729.

Ramadge, P.J.G. and Wonham, W.M. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77(1), 81–98.

Su, R. (2018). Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations. *Automatica*, 94, 35–44.

Thorsley, D. and Teneketzis, D. (2006). Intrusion detection in controlled discrete event systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 6047–6054. San Diego, CA, USA.

Wakaiki, M., Tabuada, P., and Hespanha, J.P. (2019). Supervisory control of discrete-event systems under attacks. *Dynamic Games and Applications*, 9(4), 965–983.

Wang, Y. and Pajic, M. (2019). Supervisory control of discrete event systems in the presence of sensor and actuator attacks. In *Proceedings of the 58th IEEE Conference on Decision and Control*, 5350–5355. Nice, France.

Yao, J., Yin, X., and Li, S. (2020). On attack mitigation in supervisory control systems: a tolerant control approach. In *Proceedings of the 59th IEEE Conference on Decision and Control*, 4504–4510. Jeju, South Korea.

You, D., Wang, S., Zhou, M., and Seatzu, C. (2022). Supervisory control of Petri nets in the presence of replacement attacks. *IEEE Transactions on Automatic Control*, 67(3), 1466–1473.

Zhang, Q., Seatzu, C., Li, Z., and Giua, A. (2021). Joint state estimation under attack of discrete event systems. *IEEE Access*, 9, 168068–168079.

Zhu, Y., Lin, L., and Su, R. (2019). Supervisor obfuscation against actuator enablement attack. In *Proceedings of the 18th European Control Conference*, 1760–1765. Napoli, Italy.