Dealing with sensor and actuator deception attacks in supervisory control

Rômulo Meira-Góes, Hervé Marchand, Stéphane Lafortune

Dealing with sensor and actuator deception attacks in supervisory control

Rômulo Meira-Góes [a], Hervé Marchand [b], Stéphane Lafortune [a]

[a] *Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA*

[b] *INRIA, Centre Rennes - Bretagne Atlantique, 35042, France*

Abstract

We consider feedback control systems where sensor readings and actuator commands may be compromised by an attacker intending to damage the system. We study this problem at the supervisory layer of the control system, using discrete event systems techniques. The attacker can *edit* the outputs from the sensors of the system before they reach the supervisory controller as well as it can *edit* actuator commands before they reach the system. In this context, we formulate the problem of synthesizing a supervisor that is *robust* against a large class of edit attacks on the sensor readings and actuator commands. Intuitively, we search for a supervisor that guarantees the safety of the system even when sensor readings and actuator commands are compromised. Given the similarities of the investigated problem to the standard supervisory control problem, our solution methodology reduces the problem of synthesizing a robust supervisor against deception attacks to a supervisory control problem. This new and intuitive solution methodology improves upon prior work on this topic.

Key words: Discrete Event Systems; Supervisory Control; Deception Attacks; Cyber-Security.

1 Introduction

Security concerns are a subject of increasing attention in the control community, e.g., see (Cardenas *et al.* 2008, Teixeira *et al.* 2012). It was only recently that security aspects started to be incorporated into the design of feedback control systems. Understanding and designing feedback control systems that are robust against attacks is of critical importance nowadays.

In this paper, we assume that the underlying uncontrolled system has been abstracted as a discrete transition system (the *plant* in this work), where sensor outputs belong to a finite set of (observable) events. A high-level supervisory controller, or simply *supervisor*, driven by observable events controls the behavior of the plant via actuator commands. Based on this event-driven model, we incorporate an attacker that hijacks a subset of the observable events and actuator commands. As depicted in Fig. 1, the attacker infiltrates and manipulates both communication channels between the plant and the supervisor; this type of attack is known as a *deception attack*. We study the design of feedback control systems that are robust against *deception attacks* at their supervisory layer. Conceptually, we answer the following question:

Can we find a supervisor that guarantees the safety of the plant even in the presence of attacks on both communication channels?

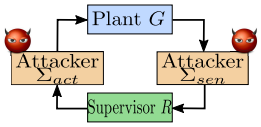


Fig. 1. Deception attack framework

Our solution methodology comprises two steps and employs techniques from supervisory control of partially-observed discrete event systems. In the first step, we build an *augmented plant*, called attacked plant, where the possible attacker's sensor manipulations are intertwined with the plant model. The attacked plant can be built in a manner that accounts for all possible attacks or can be built based on a known attack model. The second step poses a supervisory control problem for the attacked plant under a safety specification and incorporates the possible attacker's actuator manipulations. This *supervisory control problem* becomes an instance of supervisory control with *arbitrary control patterns* under partial observation, for which the existing theory of supervisory control of discrete event systems is leveraged (Golaszewski and Ramadge 1987, Li

⋆ RMG and SL are supported in part by the US National Science Foundation under grants CNS-1738103 and CNS-1801342.

Email addresses: romulo@umich.edu (Rômulo Meira-Góes), herve.marchand@inria.fr (Hervé Marchand), stephane@umich.edu (Stéphane Lafortune).

et al. 1998, Takai 2000). We show that the solution of the supervisory control problem for the attacked plant provides a solution for the problem addressed in this paper. Moreover, our solution methodology using control patterns allows a separation between the sensor and actuator attack constraints, i.e., how these attacks affect the plant and the supervisor.

Related work

Prior work on deception attacks in the field of Discrete Event Systems (DES) mainly focuses on three axes: attack detection (Thorsley and Teneketzis 2006, Carvalho *et al.* 2018, Lima *et al.* 2018, Lima *et al.* 2019), attack synthesis (Su 2018, Zhang *et al.* 2018, Lin *et al.* 2019*a*, Meira-Góes *et al.* 2020, Meira-Góes *et al.* 2021*c*), and robust supervisor synthesis (Wakaiki *et al.* 2018, Su 2018, Meira-Góes *et al.* 2019, Lin *et al.* 2019*b*, Zhu *et al.* 2019, Wang and Pajic 2019). Attack detection focuses on identifying conditions for attack detection for a given closed-loop system. Attack synthesis problems investigate the automatic design of successful attack strategies for a given supervisory control system. Lastly, robust supervisor synthesis problems against attacks study techniques to synthesize supervisors that are robust-by-construction.

Our work differs intrinsically from attack detection and attack synthesis since it focuses on robust supervisor synthesis. Each work investigating the robust supervisor synthesis problem focuses on different attack constraints. For example, robustness against bounded sensor deception attacks is studied in (Su 2018); robustness against parameterized sensor deception attacks is considered in (Meira-Góes *et al.* 2019, Meira-Góes *et al.* 2021*d*); robustness against actuator and sensor-replacement deception attacks is explored in (Lin *et al.* 2019*b*); and robustness against actuator and sensor deception attacks is investigated in (Wang and Pajic 2019). The works in (Su 2018, Wakaiki *et al.* 2018, Meira-Góes *et al.* 2019, Meira-Góes *et al.* 2021*d*) only focus on sensor deception attacks.

Compared to (Lin *et al.* 2019*b*), our work complements their results by providing a general solution for the problem of synthesizing robust supervisors against both actuator and sensor deception attacks. Sensor deception attacks in (Lin *et al.* 2019*b*) only encompass *sensor replacement attacks*, i.e., the attacker either replaces a compromised sensor reading by another reading or deletes the sensor reading. In our work, sensor deception attacks are parameterized based on prior attack knowledge, i.e., bounded attacks, replacement attacks, etc. However, we leave for future work parameterizing actuator deceptions attacks, which is possible in (Lin *et al.* 2019*b*). Lastly, our solution approach reduces the robust supervisor synthesis problem to a supervisory control problem, which provides a more intuitive solution in our opinion.

The supervisory control framework in (Wang and Pajic 2019) differs from the standard framework since they assume that the supervisor can *actively* change the state of the physical process. Therefore, our results also differ from the ones in (Wang and Pajic 2019) since we study the robustness problem in the standard supervisory control framework.

Contributions

We have four main contributions in this paper: (i) using the idea of control patterns, we describe a new modeling technique to define the closed-loop behavior of supervisory control systems under sensor and actuator attacks; (ii) we reduce the robust supervisor synthesis problem against both sensor and actuator attacks to an instance of the supervisor synthesis problem under *arbitrary control patterns* constraints; (iii) we provide sufficient conditions on the existence of the supremal robust supervisor against sensor and deception attacks; and (iv) we discuss necessary modifications on supervisory control synthesis algorithms (Hadj-Alouane *et al.* 1996, Yin and Lafortune 2016) to synthesize maximal robust supervisors and all robust supervisors. Preliminary results of this paper appeared in (Meira-Góes *et al.* 2019), where only sensor deception attacks were considered. Herein, we extend our previous work by considering both sensor and actuator attacks as well as providing the missing proofs in (Meira-Góes *et al.* 2019).

Organization

This paper is organized as follows. Section 2 introduces necessary background used throughout the paper. The framework of supervisory control theory under sensor and actuator deception attacks and the synthesis of robust supervisors are presented in Section 3. In Section 4, we provide the solution methodology for the studied problem and discuss its correctness. Sections 5 discusses two different algorithms that synthesize robust supervisors and the trade-off between them. We conclude the paper in Section 6.

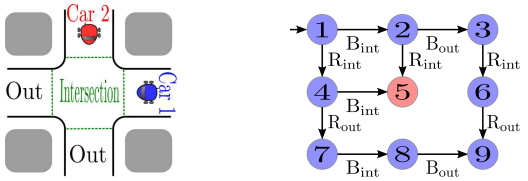
2 Preliminaries

We consider systems modeled as automata. Mathematically, an automaton G is defined as a tuple $G := (X_G, \Sigma, \delta_G, x_{0,G})$, where X_G is a finite set of states; Σ is a finite set of events; $\delta_G : X_G \times \Sigma \to X_G$ is a partial transition function; and $x_{0,G} \in X_G$ is the initial state. The function δ_G is extended in the usual manner to domain $X_G \times \Sigma^*$, where $*$ is the standard Kleene star operation. The language generated by G is defined as $\mathcal{L}(G) := \{s \in \Sigma^* | \delta_G(x_0, s)!\}$, where ! means "is defined". For $x \in X_G$, we define $En_G(x) := \{e \in \Sigma \mid \delta_G(x, e)!\}$ as the active event set at state x. For $K \subseteq \Sigma^*$, we denote by $pr(K)$ the set of all prefixes of strings in K and K is said to be prefix-closed if $K = pr(K)$.

In the context of supervisory control theory of DES (Ramadge and Wonham 1987), an uncontrolled system (plant) G is controlled by a supervisor, denoted by S. This controlled system is a new DES denoted by S/G with closed-loop language $\mathcal{L}(S/G)$ defined in the usual manner (see, e.g., (Cassandras and Lafortune 2008, Wonham and Cai 2018)). Usually, the supervisor S has limited actuation and sensing capabilities described by the pair of partitions of Σ (Σ_c, Σ_{uc}) and (Σ_o, Σ_{uo}), which represent the sets of controllable, uncontrollable, observable, and unobservable events, respectively. Intuitively, the supervisor only observes events in Σ_o and only disables events

in Σ_c. Formally, it is a function $S : P_o(\mathcal{L}(G)) \to \Gamma$, where P_o is the natural event projection from Σ^* to Σ_o^* extended to sets, and the set Γ is the set of admissible control decisions, i.e., $\Gamma := \{\gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma\}$. Without loss of generality, we assume that S is realized by an automaton $R = (X_R, \Sigma, \delta_R, x_{0,R})$ (see, e.g., (Cassandras and Lafortune 2008, Wonham and Cai 2018)). While the domain of events in R is Σ not Σ_o, its transitions are only driven by *observable* events, i.e., transitions with unobservable events are self-loops. We use both notations S and R interchangeably throughout the paper.

Example 1 *We model the road intersection example depicted in Fig. 2(a), which will be used as a simple illustrative running example hereafter. The cars must cross the intersection without colliding with each other, or equivalently, both cannot be at the intersection at the same time. This system is modeled by the automaton G shown in Fig. 2(b). Every event is controllable and observable. The supervisor realization R that guarantees the above specification is obtained by deleting state 5.*



(a) Intersection with two cars

(b) Road intersection model. Events with B(lue) are related to car 1, while events with R(ed) are related to car 2.

Fig. 2. Intersection example

3 Robust Supervisory Control against Deception Attacks

3.1 Supervisory control under sensor deception attacks

Figure 1 pictorially describes deception attacks in the supervisory control framework, where the attacker intervenes in both communication channels. In the communication channel between the plant and the supervisor, the attacker has the ability to observe the same observable events as the supervisor. Even more, it has the ability to alter some of these observed events, where "alter" means that it can insert or delete events. The subset of affected sensor readings is denoted by $\Sigma_{sen} \subseteq \Sigma_o$, where only observable events can be compromised since unobservable events have no effect on the supervisor.

In the communication channel between the supervisor and the plant, the attacker has the ability of enabling/disabling some of the controllable events that are disabled/enabled by the current control decision. The subset of affected controllable events is denoted by $\Sigma_{act} \subseteq \Sigma_c$, where only controllable events can be compromised since uncontrollable events are always enabled by the supervisor.

First, we must characterize the closed-loop behavior of this controlled system under deception attacks. After this characterization, we will formalize robustness against deception attacks. Our characterization follows similar lines as in (Carvalho *et al.* 2018, Lima *et al.* 2018). The conceptual diagram of deception attacks, as shown in Fig. 1, is transformed into an attacked controlled system depicted in Fig. 3. This attacked system is constructed based on the original controlled system and attacker assumptions.

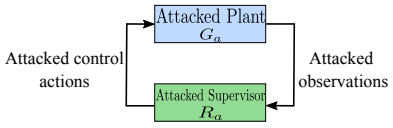


Fig. 3. Attacked Controlled System

Intuitively, the attacked plant is constructed by intertwining the possible attacker's sensor manipulations with the plant model. On the other hand, the attacked supervisor is built based on these possible sensor modifications as well as incorporating possible event enablements by the attacker. Combining both models characterize the closed-loop behavior under deception attacks.

3.2 Closed-loop system under deception attacks

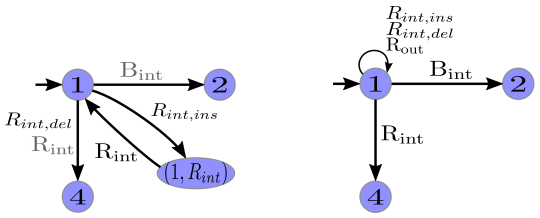
3.2.1 Attacked plant

As was mentioned above, the attacked plant is constructed by intertwining the possible attacker's sensor manipulations with the plant model. Let us look at a concrete example of the construction of the attacked plant using G from Example 1 and $\Sigma_{sen} := \{R_{int}\}$. If we assume that the plant is at its initial state, state 1, then the plant can execute events B_{int} or R_{int}. Since we combine the attacker with the plant actions, the attacker can insert event R_{int} in the communication channel before the plant executes event B_{int} or R_{int}. Lastly, when the plant executes R_{int}, the attacker can delete this event from the communication channel.

Figure 4(a) depicts the above described transitions in state 1 of the attacked plant. Insertions are encoded by two transitions: transition from 1 to $(1, R_{int})$ with event $R_{int,ins}$; and transition from $(1, R_{int})$ to 1 with event R_{int}. This encoding simulates the insertion of event R_{int} in state 1 since the plant remains in the same state after the insertion. The sequence $R_{int,ins}R_{int}$ uniquely identifies the insertion of event R_{int} in the attacked plant, i.e., event $R_{int,ins}$ signals that the following event R_{int} is inserted by the attacker. The reasoning behind this encoding becomes clear in the closed-loop system as we will discuss it later. Intuitively, event $R_{int,ins}$ is defined to be unobservable, i.e., the supervisor cannot observe $R_{int,ins}$. If the attacked plant executes the sequence $R_{int,ins}R_{int}$, the plant remains in state 1 while the supervisor will receive observation R_{int}. As to ensure the correct encoding of this behavior, we must introduce the new state $(1, R_{int})$ to enforce the execution of event R_{int} after event $R_{int,ins}$ occurs.

On the other hand, event $R_{int,del}$ identifies the execution and deletion of event R_{int}. The deletion event is defined in parallel to transition R_{int} since event R_{int} must be executed by G in order to be deleted from the communication

channel. Looking at the closed-loop system, event $R_{int,del}$ will also be unobservable to the supervisor. Therefore, if the attacked plant executes event $R_{int,del}$, the plant moves to state 4 while the supervisor will not receive any information. Moreover, note that the attacked plant generates more behavior than the original plant since the attacked plant combines attacker actions with legitimate plant executions. Nevertheless, the attacked plant correctly captures the state evolution of the original plant while combining the attacker actions with legitimate plant executions.



(a) Def. of δ_{G_a} for state 1 (b) Def. of δ_{R_a} for state 1

Fig. 4. Attacked plant and attacked supervisor

To formalize the definition of the attacked plant, we need to distinguish the attack actions from legitimate events generated by G, e.g., $R_{int,ins}$ and $R_{int,del}$. Thus, we define the set of attacked events $\Sigma_{att} := \Sigma_{ins} \cup \Sigma_{del}$, where $\Sigma_{ins} := \{e_{ins} \mid e \in \Sigma_{sen}\}$ and $\Sigma_{del} := \{e_{del} \mid e \in \Sigma_{sen}\}$ denote insertion and deletion events, respectively. These sets are pairwise disjoint with Σ_{sen}. We also define the operator $\mathcal{M}$ that removes the ins and del subscripts from events: $\mathcal{M}(e_{ins}) = \mathcal{M}(e_{del}) := e$ for all $e \in \Sigma_{sen}$ and $\mathcal{M}(e) := e$ for all $e \in \Sigma$.

Based on Σ_{att}, we augment the plant G with Σ_{att} to create the attacked plant G_a, i.e., G_a is a copy of G with additional events Σ_{att} and additional states. In the attacked plant G_a, events e_{ins} simulate the insertion ability of the attacker, whereas events e_{del} simulate the deletion ability of the attacker. Formally, G_a is defined as follow:

Definition 2 *Given G and Σ_{sen}, we define $G_a := (X_{G_a}, \Sigma_m, \delta_{G_a}, x_{0,G_a})$, where $X_{G_a} := X_G \cup (X_G \times \Sigma_{sen})$, $\Sigma_m := \Sigma \cup \Sigma_{att}$, $x_{0,G_a} := x_{0,G}$ and*

$$\delta_{G_a}(x,e) := \begin{cases} \delta_G(x, \mathcal{M}(e)) & \text{if } x \in X_G \text{ and } e \notin \Sigma_{ins} \\ (x, \mathcal{M}(e)) & \text{if } x \in X_G \text{ and } e \in \Sigma_{ins} \\ x_1 & \text{if } x = (x_1, e_1),\ e = e_1 \\ \text{undefined} & \text{otherwise} \end{cases} \tag{1}$$

To complete our encoding of G_a, the unobservable event set in G_a is defined as $\Sigma_{m,uo} := \Sigma_{uo} \cup \Sigma_{att}$. Unobservable events in G continue to be unobservable in G_a. The attacker actions Σ_{att} are also considered unobservable in our encoding since the supervisor does not observe them. Insertions events e_{ins} can be considered unobservable since they are immediately followed by the observable event $\mathcal{M}(e_{ins}) = e$ in G_a, e.g., $1 \xrightarrow{R_{int,ins}} (1, R_{int}) \xrightarrow{R_{int}} 1$ in Fig. 4(a) . Finally, the natural string projection function $P_{m,o}$ projects strings in Σ_m^* to Σ_o^*, i.e., $P_{m,o} : \Sigma_m^* \to \Sigma_o^*$.

3.2.2 *Attacked supervisor*

The construction of the attacked supervisor is similar to the construction of the attacked plant. However, we combine the possible attacker's *actuator* manipulations with the supervisor model. Moreover, the attacked supervisor is constructed for the attacked plant, which contains attacked events Σ_{att}. For this reason, the attacked supervisor must include the set of attacked events Σ_{att} defined in the previous section. First, we define *the set of control patterns* that a supervisor for G_a must follow. This set is defined based on the set of compromised actuator events Σ_{act} and the attacked event Σ_{att}. Next, we augment the supervisor R based on this new set of control patterns to obtain the attacked supervisor R_a.

As was defined in Section 3.1, the events in Σ_{act} are the compromised actuator events, which the attacker can enable or disable at any given point. Disabling events can be advantageous for the attacker to better steer the plant to desired states, but it can only reduce the overall controlled behavior, i.e., it reduces the set of reachable states. Therefore, for the problem of synthesizing robust supervisors, it is sufficient to only consider actuator enablements (Carvalho *et al.* 2018). In this manner, events in Σ_{act} must be deemed as uncontrollable since the attacker can arbitrarily enable them.

Next, we introduce attacked events Σ_{att} into the attacked supervisor model. These events cannot be directly controlled by the supervisor since the attacker decides when to attack. However, if a compromised event $e \in \Sigma_{sen}$ is controllable, then the supervisor can indirectly control the events in Σ_{att}. For example, if event R_{int} is disabled when the plant is in state 1, then the event $R_{int,del}$ is also disabled since event R_{int} will not be executed. In the case of insertion events, for convenience and without loss of generality, we state the following assumption.

Assumption 3 *The attacked supervisor enables insertion events if and only if it enables the corresponding legitimate event.*

We could have assumed that the insertion of a disabled event is simply ignored by the supervisor, i.e., it does not change its state. Since inserting events also does not change the state of the plant (Def. 2), Assumption 3 does not reduce the power of the attacker, i.e., it is without loss of generality. Based on this assumption, the insertion of event R_{int} also depends on the enablement of event R_{int}. Therefore, if event $e \in \Sigma_{sen}$ is enabled, then it consequently enables events e_{ins} and e_{del}. Based on the above discussion, *the control patterns* C_a is defined as:

$$\begin{aligned} C_a = \{&\gamma \in 2^{\Sigma_m} \mid \Sigma_{uc} \cup \Sigma_{act} \subseteq \gamma \wedge \\ &((e \in \gamma \cap \Sigma_{sen}) \Leftrightarrow (e_{ins}, e_{del} \in \gamma))\} \end{aligned} \tag{2}$$

The condition $\Sigma_{uc} \cup \Sigma_{act} \subseteq \gamma$ implies that events in Σ_{act} are treated as uncontrollable events. The second condition states that enabling an event in Σ_{sen} implies enabling its

insertion and deletion. Note that the set C_a is closed under union, i.e., if $\gamma_1, \gamma_2 \in C_a$, then $\gamma_1 \cup \gamma_2 \in C_a$. The attacked supervisor R_a is constructed such that it only emits control decisions in C_a.

Definition 4 *Given R, Σ_{att} and Σ_{act}, we define the attacked supervisor $R_a := (X_{R_a}, \Sigma_m, \delta_{R_a}, x_{0,R_a})$, where $X_{R_a} := X_R$, $x_{0,R_a} := x_{0,R}$, and for $e \in \Sigma_m$*

$$\delta_{R_a}(x,e) := \begin{cases} \delta_R(x, P_{m,o}(e)) & \text{if } \mathcal{M}(e) \in En_R(x) \\ x & \text{if } \mathcal{M}(e) \in \Sigma_{act} \setminus En_R(x) \\ \text{undefined} & \text{otherwise} \end{cases} \tag{3}$$

Let us look at a concrete example of the construction of the attacked supervisor using R from Example 1, $\Sigma_{sen} := \{R_{int}\}$, and $\Sigma_{act} = \{R_{out}\}$. Figure 4(b) depicts all transitions defined by Eq. (3) in state 1 of the attacked supervisor. The first condition in Eq. (3) states that if a compromised event $e \in \Sigma_{sen}$ is enabled in state $x \in X_R$, then e, e_{ins}, and e_{del} are enabled in $x \in X_{R_a}$. In Fig. 4(b), events $R_{int,ins}, R_{int,del}$ are defined as self-loops in state 1, i.e., $\delta_{R_a}(1, R_{int,ins}) = \delta_R(1, P_{m,o}(R_{int,ins})) = \delta_R(1, \epsilon) = 1$, On the other hand, event R_{int} in state 1 is defined as in R, i.e., $\delta_{R_a}(1, R_{int}) = \delta_R(1, P_{m,o}(R_{int})) = \delta_R(1, R_{int}) = 4$. If the event is not compromised, then only this event is enabled in R_a, as for event B_{int} in Fig. 4(b). The second condition in Eq. (3) guarantees that compromised actuator events are always enabled in R_a, i.e., event R_{out} is defined as a self-loop in state 1 of the attacked supervisor. In this manner, supervisor R_a is a copy of R with additional self-loops based on Σ_{att} and Σ_{act}.

3.2.3 Closed-loop system

The definition of the control pattern set C_a goes hand in hand with the attacked plant encoding. To execute the sequence $R_{int,ins}R_{int}$, the supervisor enables events $R_{int,ins}$ and R_{int} (State 1 in Fig. 4b). After the execution, the supervisor moves to state 2 in Fig. 4b, i.e., as if it had observed event R_{int}. On the other hand, the attacked plant goes through state $(1, R_{int})$ and reaches state 1 after the sequence $R_{int,ins}R_{int}$ in Fig. 4a, i.e., the plant remains in state 1 after the attacker insertion. Event deletion happens in a similar way.

In this manner, the closed-loop system under deception attacks is captured by the parallel composition of $G_a||R_a$, where $||$ is the parallel composition operator as in (Cassandras and Lafortune 2008). Since R_a is a supervisor realization, the closed-loop system under deception attacks is also denoted by R_a/G_a. The language $\mathcal{L}(R_a/G_a)$ generated by this system is defined as the usual closed-loop language of the supervised system.

Remark 5 *The definitions of G_a and R_a are based on the worst-case attack scenario in which the attacker* can *attack whenever it is possible. This attack strategy introduced in (Carvalho* et al. *2018, Lima* et al. *2018) is called the* "all-out" *attack strategy. Although this model is simple, it is well-suited to the problem we are investigating since we want to design a supervisor that is robust against any deception attack. At the end of this section, we generalize our closed-loop behavior based on prior attack knowledge.*

3.3 Supervisor robust against deception attacks

We assume that the plant G contains a set of *critical* states defined as $X_{crit} \subset X_G$; these states are unsafe in the sense that they are states where damage to the plant might occur. Although damage is defined in relation to the set X_{crit}, it could be generalized in relation to any regular language by state space refinement in the usual way (Cho and Marcus 1989, Cassandras and Lafortune 2008). Since $X_G \subset X_{G_a}$ and with an abuse of notation, we say that X_{crit} defines the critical states of G_a.

Definition 6 *Given plant G, X_{crit}, supervisor R, and sets $\Sigma_{sen}, \Sigma_{act}$, we say that R is robust with respect to G and $(\Sigma_{act}, \Sigma_{sen})$ if $\delta_{G_a}(x_0, s) \notin X_{crit}$ for any $s \in \mathcal{L}(R_a/G_a)$.*

Finally, we are able to formally pose the Synthesis of Robust Supervisor Problem.

Problem 7 (RSS–Robust Supervisor Synthesis) *Given plant G, X_{crit}, and sets $\Sigma_{sen} \subseteq \Sigma_o$, $\Sigma_{act} \subseteq \Sigma_c$, synthesize a robust supervisor R with respect to $(\Sigma_{act}, \Sigma_{sen})$, if one exists.*

Example 8 *Let us return to our intersection example, where we assume that $\Sigma_{sen} = \{R_{int}\}$ and $\Sigma_{act} = \{R_{out}\}$. Using G and R as given in Example 1, we construct the pair G_a and R_a. Based on them, we obtain the attacked system R_a/G_a shown in Fig. 5. The string $s = R_{int,del}B_{int}$, in red in Fig. 5, is feasible in R_a/G_a and takes G_a to state 5. Therefore, this supervisor is not robust with respect to $\Sigma_{sen} = \{R_{int}\}$ and $\Sigma_{act} = \{R_{out}\}$.*

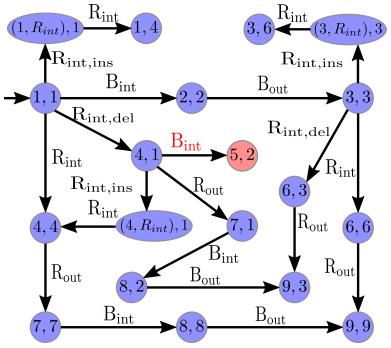


Fig. 5. The attacked controlled system for R_a/G_a

3.4 Prior attack knowledge

In the previous section, we defined the attacked behavior based on the "all-out" attack strategy. We now relax this assumption based on some prior knowledge of the attack strategy used by an attacker. We only restrict the attacker based on a sensor deception attack strategy, and we leave restricting actuator attacks for future work. Using such knowledge, we can build the attacked plant similarly as it was built for the "all-out" attack strategy. Namely, we need to modify the construction of the attacked plant G_a. However, the construction of R_a remains the same as before.

There are different ways of describing attack strategies. We assume that the known attack strategy is encoded as an automaton $A = (X_A, \delta_A, \Sigma_m, x_{0,A})$ as in (Meira-Góes *et al.* 2020, Meira-Góes *et al.* 2021*d*). If automaton A correctly encodes the attack strategy, then we can restrict G_a as defined in Def. 2 based on A, i.e., $G'_a := G_a||A$. In this manner, the attacked system is defined by $G'_a := G_a||A$ and R_a is as in Def. 4.

We give one example of an attack strategy to demonstrate how G'_a is constructed. The automaton in Fig. 6 encodes an attack strategy where the attacker deletes at most one event. The attacker is initialized in state 1, where it has not performed any deletion. In state 1, the attacker can observe legitimate events (self-loop with Σ) or it can delete an event (transition $1 \xrightarrow{\Sigma_{del}} 2$). Once the attacker deletes one event, it moves to state 2, where it only observes legitimate events.

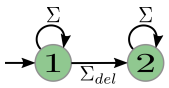


Fig. 6. One deletion attack strategy

In the rest of the paper, we present our results for G_a as defined in Def. 2. However, these results are applicable to G'_a constructed based on a known attack strategy A by simply replacing G_a by G'_a.

Remark 9 *In (Lin* et al. *2019*b*), attack strategies are constrained based on prior knowledge of the attack strategy. However, their techniques to constrain actuator attacks do not apply to our framework since they reduce the robustness problem to a quantified Boolean formula problem. Moreover, sensor attacks are restrained to sensor* replacement *attacks.*

4 Solution of the Robust Supervisory Control Problem

4.1 Supervisory control with arbitrary control patterns

As described in the definition of the attacked supervisor, the supervisor for the attacked plant must select its control decision based on the set of control patterns C_a, i.e., a supervisor for the attacked plant is defined as $S : \Sigma_o^* \to C_a$. This supervisor is more constrained compared to the general supervisor defined in Section 2, in which the supervisor leverages the entire set of feasible control decisions. For this reason, prior results from supervisory control theory with arbitrary control patterns are essential for our proposed framework (Golaszewski and Ramadge 1987, Li *et al.* 1998, Takai 2000). We state these results for the attacked plant G_a and the set of control patterns C_a.

Proposition 10 *(Takai 2000) Let $K \subseteq \mathcal{L}(G_a)$ be a non-empty and prefix-closed language and let C_a be the available control patterns set. There exists a supervisor $S : P_{m,o}(\mathcal{L}(G_a)) \to C_a$ such that $\mathcal{L}(S/G_a) = K$ if and only if K satisfies the following condition:*

$$(\forall s \in K)(\exists \gamma \in C_a)(\forall t \in P_{m,o}^{-1}[P_{m,o}(s)] \cap K) \text{ s.t. } \gamma \cap \Sigma_{\mathcal{L}(G_a)}(t) = \Sigma_K(t)$$

where $P_{m,o}^{-1}$ is the inverse projection operator and $\Sigma_K(s) := \{e \in \Sigma_m \mid se \in K\}$ is the one event continuation from $s \in K$.

Proposition 10 reduces to the standard controllability and observability conditions when $C_a = \{\gamma \in 2^{\Sigma_m} \mid \Sigma_{uc} \cup \Sigma_{act} \subseteq \gamma\}$ (Takai 2000, Cassandras and Lafortune 2008). When K does not satisfy Proposition 10 given the set C_a, the set $\mathcal{CO}(K, C_a) = \{K' \subseteq \mathcal{L}(G_a) \mid K' = pr(K') \subseteq K \text{ s.t. } K' \text{ satisfies Proposition 10}\}$ is defined. This set is non-empty since $\emptyset \in \mathcal{CO}(K, C_a)$. As in the standard partial observation supervisory control problem, there does not exist in general a supremal element in $\mathcal{CO}(K, C_a)$ (Takai 2000).

4.2 Existence of solution for the RSS problem

Our definition of robustness against deception attacks, Def. 6, focuses exclusively on the language $\mathcal{L}(R_a/G_a)$, which is the language of the attacked system. Therefore, this language becomes the center of our study, where we search for a supervisor for the attacked plant G_a such that the controlled attack system satisfies the robustness property. Here, we investigate G_a in the supervisory control framework with arbitrary control patterns C_a and observable event set Σ_o.

Let $K_{spec} := \mathcal{L}(Ac(G_a, X_{crit}))$ be the language specification on $\mathcal{L}(G_a)$, where $Ac(G_a, X_{crit})$ is the *accessible* subautomaton of G_a after deleting states $X_{crit} \subset X_{G_a}$. From (Takai 2000), it follows that for any $K' \in \mathcal{CO}(K_{spec}, C_a)$, if $K' \neq \emptyset$, there exists a supervisor $S : P_{m,o}(\mathcal{L}(G_a)) \to C_a$ such that $\mathcal{L}(S/G_a) = K'$. Since the languages $\mathcal{L}(G_a)$ and K_{spec} are regular languages, without loss of generality we assume that the supervisor $S : P_{m,o}(\mathcal{L}(G_a)) \to C_a$ such that $\mathcal{L}(S/G_a) \in \mathcal{CO}(K_{spec}, C_a)$ is encoded by a DFA R_a.

Definition 11 *Let the supervisor R_a satisfy $\mathcal{L}(R_a/G_a) \in \mathcal{CO}(K_{spec}, C_a)$. Build supervisor $R := (X_R, \Sigma, \delta_R, x_{0,R})$, where $X_R := X_{R_a}$, $x_{0,R} := x_{0,R_a}$, and $\delta_R(x, e) := \delta_{R_a}(x, e)$ for $e \in \Sigma$ and $x \in X_R$, i.e., δ_R is δ_{R_a} restricted to events in Σ.*

Definition 11 constructs the supervisor R by simply removing events in Σ_{att} from R_a. The transitions with events in Σ_{att} on R_a are self-loops since they are unobservable in the attacked plant G_a. Therefore, Def. 11 is the "inverse construction" of Def. 4.

Robust supervisors with respect to G, X_{crit}, and $(\Sigma_{act}, \Sigma_{sen})$ are directly related to supervisors R_a such that $\mathcal{L}(R_a/G_a) \in \mathcal{CO}(K_{spec}, C_a)$. Intuitively, if a supervisor R is robust, then the language $\mathcal{L}(R_a/G_a) \in \mathcal{CO}(K_{spec}, C_a)$, where R_a is constructed via Def. 4. On the other hand, let R_a be a supervisor such that $\mathcal{L}(R_a/G_a) \in \mathcal{CO}(K_{spec}, C_a)$, then the supervisor R constructed via Def. 11 is robust. The following theorem formally states this connection.

Theorem 12 *A supervisor R is robust w.r.t. G, X_{crit}, and $(\Sigma_{act}, \Sigma_{sen})$ if and only if $\mathcal{L}(R_a/G_a) \in \mathcal{CO}(K_{spec}, C_a)$.*

Proof. We start with the only if part. Assuming that R is robust w.r.t. G and $(\Sigma_{act}, \Sigma_{sen})$, we need to show that $\mathcal{L}(R_a/G_a) \in \mathcal{CO}(K_{spec}, C_a)$, where R_a is constructed via Def. 4. In other words, we need to show that

$\mathcal{L}(R_a/G_a) \subseteq K_{spec}$ and satisfies Prop. 10. We show this result directly by constructing the language $\mathcal{L}(R_a/G_a)$. Language $K^* := \mathcal{L}(R_a/G_a)$ is a subset of K_{spec} since R is robust w.r.t. $(\Sigma_{act}, \Sigma_{sen})$, i.e., no string in K^* reaches the critical states. Next, every control decision in R_a satisfies by construction the set of control patterns in C_a. We just need to show that for any $s \in K^*$, there exists $\gamma \in C_a$ such that $\gamma \cap \Sigma_{\mathcal{L}(G_a)}(t) = \Sigma_{K^*}(t)$ for any $t \in P_{m,o}^{-1}[P_{m,o}(s)] \cap K^*$ (as stated in Prop. 10). Let $\gamma := En_{R_a}(\delta_{R_a}(x_{0,R_a}, P_{m,o}(t)))$ so that we compare both sides of the equation. Note that, $\delta_{R_a}(x_{0,R_a}, P_{m,o}(t))$ is well defined since $s \in K^*$ and $P_{m,o}(s) = P_{m,o}(t)$. In the right hand side, by definition of the controlled language $\mathcal{L}(R_a/G_a)$, $e \in \Sigma_{K^*}(t)$ if and only if $t \in K^* = \mathcal{L}(R_a/G_a)$, $e \in \gamma$, and $te \in \mathcal{L}(G_a)$. Since we assumed that $t \in P_{m,o}^{-1}[P_{m,o}(s)] \cap K^*$, then $e \in \Sigma_{K^*}(t)$ if and only if $e \in \gamma$ and $te \in \mathcal{L}(G_a)$. On the left hand side, $e \in \gamma \cap \Sigma_{\mathcal{L}(G_a)}(t)$ if and only if $t \in \mathcal{L}(G_a)$, $e \in \gamma$, and $te \in \mathcal{L}(G_a)$. Again, since we assumed that $t \in P_{m,o}^{-1}[P_{m,o}(s)] \cap K^* \subseteq \mathcal{L}(G_a)$, then $e \in \gamma \cap \Sigma_{\mathcal{L}(G_a)}$ if and only if $e \in \gamma$ and $te \in \mathcal{L}(G_a)$. It follows that $\gamma \cap \Sigma_{\mathcal{L}(G_a)}(t) = \Sigma_{K^*}(t)$.

We now show the if part. We assume that $\mathcal{L}(R_a/G_a) \in \mathcal{CO}(K_{spec}, C_a)$ and prove that R constructed via Def. 11 is robust with respect to $(\Sigma_{act}, \Sigma_{sen})$. By definition of $\mathcal{CO}(K_{spec}, C_a)$, it follows that of $\mathcal{L}(R_a/G_a) \subseteq K_{spec}$. Since $\mathcal{L}(R_a/G_a) \subseteq K_{spec}$, the critical states in G_a are unreachable by any $s \in \mathcal{L}(R_a/G_a)$. Therefore, R is a robust supervisor with respect to $(\Sigma_{act}, \Sigma_{sen})$. □

Theorem 12 provides a set of sufficient and necessary conditions for existence of a solution for our **RSS** problem. In words, there exists a robust supervisor if and only if there exists a supervisor R_a such that $\mathcal{L}(R_a/G_a) \in \mathcal{CO}(K_{spec}, C_a)$. Corollary 13 clearly states this result.

Corollary 13 *There exists a solution for the **RSS** problem if and only if $\mathcal{CO}(K_{spec}, C_a) \neq \{\emptyset\}$.*

Theorem 12 also states one way of obtaining a solution for the **RSS** problem, when one exists. A solution can be obtained by constructing the set $\mathcal{CO}(K_{spec}, C_a)$ or constructing specific elements in $\mathcal{CO}(K_{spec}, C_a)$. However, no algorithm to construct supervisors in $\mathcal{CO}(K_{spec}, C_a)$ is described in (Takai 2000). For this reason, we discuss two methods to obtain solutions of the **RSS** problem based on the set $\mathcal{CO}(K_{spec}, C_a)$ in Section 5.

4.3 Sufficient condition for the existence of the supremal robust supervisor

In the previous sections, we showed how to solve the general problem of synthesizing robust supervisors. Since there does not exist in general a supremal supervisor for this problem, we investigate sufficient conditions for the existence of such a supremal robust supervisor.

The first condition, also known as normality condition, is well known in the partially observed supervisory control problem: $\Sigma_c \subseteq \Sigma_o$. When $\Sigma_c \subseteq \Sigma_o$, it follows that the observability plus the controllability conditions imply the normality condition (Cassandras and Lafortune 2008). Since the supremal controllable and normal sublanguage always exists, then, with this condition imposed, the supremal controllable and observable sublanguage also exists. Recall that, in the case of G_a, the events in Σ_{act} are treated as uncontrollable. Therefore, the above condition is $\Sigma_c \setminus \Sigma_{act} \subseteq \Sigma_o$. However, this condition is not enough to guarantee the existence of the supremal robust supervisor. In fact, our running example demonstrates the nonexistence of a supremal element even when $\Sigma_c \setminus \Sigma_{act} \subseteq \Sigma_o$ as we will see in Section 5.

In cyber-physical system models, we normally assign actuators signals to be controllable and observable events while sensors signals are uncontrollable and observable. Since we study *sensor* deception attacks, let us assume that the set of affected sensor events is a subset of the uncontrollable events, besides being a subset of observable events. Thus, the second condition is $\Sigma_{sen} \subseteq \Sigma_o \cap \Sigma_{uc}$.

Theorem 14 *If $\Sigma_c \setminus \Sigma_{act} \subseteq \Sigma_o$ and $\Sigma_{sen} \subseteq \Sigma_{uc} \cap \Sigma_o$, then there exists a supremal element $K^{\uparrow} \in \mathcal{CO}(K_{spec}, C_a)$ and supervisor R_a such that $K^{\uparrow} = \mathcal{L}(R_a/G_a)$.*

5 Computing Robust Supervisors

Herein, we discuss two methods to compute robust supervisors. Due to space limitations, the complete discussion on these methods is described in (Meira-Góes *et al.* 2021*a*).

5.1 Computing a Maximal Robust Supervisor

In order to synthesize a robust supervisor, Corollary 13 states that we can select any supervisor R_a that satisfies $\mathcal{L}(R_a/G_a) \in \mathcal{CO}(K_{spec}, C_a)$. Since the supremal element of $\mathcal{CO}(K_{spec}, C_a)$ does not exist in general, we search for a supervisor that generates a maximal language in $\mathcal{CO}(K_{spec}, C_a)$. A maximal language $L \in \mathcal{CO}(K_{spec}, C_a)$ satisfies: $\nexists L' \in \mathcal{CO}(K_{spec}, C_a)$ such that $L \subset L'$.

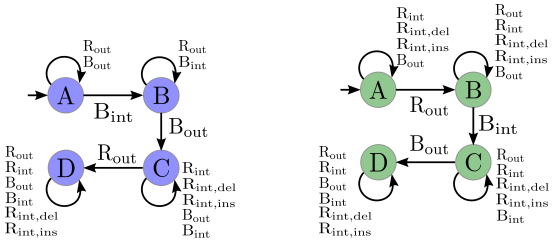
We can adapt standard algorithms that compute maximal controllable and observable sublanguages to consider control patterns. For example, the VLP-PO algorithm from (Hadj-Alouane *et al.* 1996) can be adapted such that it considers control patterns. Intuitively, the only difference between modified VLP-PO algorithm and the VLP-PO algorithm presented in (Hadj-Alouane *et al.* 1996) is that the latter selects control decisions from the set of all possible feasible control decisions, i.e., $\{\gamma \in 2^{\Sigma_m} \mid \Sigma_{uc} \cup \Sigma_{act} \subseteq \gamma\}$, where the former will select control decisions from C_a. For this reason, the modified algorithm guarantees the same results as the VLP-PO algorithm, e.g., correctness and complexity.

5.2 Finding all Robust Supervisors

Although the modified VLP-PO algorithm provides some flexibility on the computation of robust supervisors through the event ordering of the controllable events, it limits its search to computing maximal robust supervisors for each ordering. For this reason, we would like to investigate an algorithm that would provide more flexibility for synthesizing robust supervisors.

In (Yin and Lafortune 2016, Meira-Góes *et al.* 2021*b*), a uniform approach for synthesizing all state-based property-enforcing supervisors for partially observed DES is presented. Their method constructs a finite bipartite transition systems, called All Enforcement Structure (AES), which embeds all admissible supervisors that enforce a desired state-based property. Our definition of robustness, Def. 6, is a state-based property in G_a. Thus, we can alter the original AES construction to adapt it to the set C_a and obtain the "modified AES"; full details are presented in (Meira-Góes *et al.* 2021*a*). The modified AES guarantees the same results as the original AES, e.g., correctness and complexity.

Example 15 *Back to the intersection example, we calculate two maximal robust supervisors. Recall that we have defined* $\Sigma_{sen} = \{R_{int}\}$ *and* $\Sigma_{act} = \{R_{out}\}$*. We construct the attacked plant based on Def. 2, where* G_a *has* 18 *states since* G *has* 9 *states and* $\Sigma_{sen} = \{R_{int}\}$*. At this point, we want to enforce that the critical state* 5 *is not reachable, i.e., the specification is defined by automaton* $Ac(G_a, \{5\})$*. We can use either the modified VLP-PO or the modified AES algorithms to obtain these supervisors. These two supervisors are shown in Fig. 7. Note that whenever* R_{int} *is enabled, then both* $R_{int,ins}$ *and* $R_{int,del}$ *are enabled. Moreover, the event* R_{out} *is enabled in every state of both supervisors. Therefore, the control pattern constraint is satisfied. We obtain a robust supervisor for* G *by removing events* Σ_{att} *from the supervisors in Fig. 7, according to Def. 11.*



(a) Robust Supervisor R_1 (b) Robust Supervisor R_2

Fig. 7. Two robust supervisors for the intersection example

6 Conclusion

We have considered a class of problems in cyber-security where sensor readings and actuator commands in a feedback control system may be manipulated by a malicious attacker. We leverage techniques from supervisory control of partially-observed discrete event systems, with arbitrary control patterns, to develop a solution methodology to prevent damage to the system when some sensor readings and actuator commands may be edited by the attacker. We showed how this problem can be reduced to a partially observed supervisory control problem with a specific class of control patterns. Then, we provided two different algorithms to obtain these robust supervisors. In Section 3, we only introduced restrictions to sensor attacks when prior knowledge is available, but we left actuator attacks restrictions to future work. We believe that it will be possible to restrict actuator attacks using control patterns and maintain the separation between sensor and actuator constraints. Synthesizing nonblocking robust supervisors is a potential extension of our work. The problem of synthesizing nonblocking supervisors with arbitrary control patterns is still an open problem, as the results in (Takai 2000) only tackle prefix-closed specifications.

References

Cardenas, A. A., S. Amin and S. Sastry (2008). Secure control: Towards survivable cyber-physical systems. In '2008 The 28th International Conference on Distributed Computing Systems Workshops'. pp. 495–500.

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

Cassandras, C. G. and S. Lafortune (2008). *Introduction to Discrete Event Systems*. 2 edn. Springer-Verlag New York, Inc.. Secaucus, NJ, USA.

Cho, H. and S. I. Marcus (1989). 'On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation'. *Mathematics of Control, Signals and Systems* **2**(1), 47–69.

Golaszewski, C. H. and P. J. Ramadge (1987). Control of discrete event processes with forced events. In '26th IEEE Conference on Decision and Control'. Vol. 26. pp. 247–251.

Hadj-Alouane, N. B., S. Lafortune and F. Lin (1996). 'Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation'. *Discrete Event Dynamic Systems* **6**(4), 379–427.

Li, Y., F. Lin and Z. H. Lin (1998). 'A generalized framework for supervisory control of discrete event systems'. *International Journal of Intelligent Control and Systems* **2**, 139–160.

Lima, P. M., L. K. Carvalho and M. V. Moreira (2018). Detectable and undetectable network attack security of cyber-physical systems. In '14th IFAC Workshop on Discrete Event Systems WODES 2018'. Vol. 51. pp. 179 – 185.

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

Lin, L., S. Thuijsman, Y. Zhu, S. Ware, R. Su and M. Reniers (2019*a*). Synthesis of supremal successful normal actuator attackers on normal supervisors. In '2019 American Control Conference (ACC)'. pp. 5614–5619.

Lin, L., Y. Zhu and R. Su (2019*b*). Towards bounded synthesis of resilient supervisors. In '2019 IEEE 58th Conference on Decision and Control (CDC)'. pp. 7659–7664.

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

Meira-Góes, R., H. Marchand and S. Lafortune (2019). Towards resilient supervisors against sensor deception attacks. In '2019 IEEE 58th Annual Conference on Decision and Control (CDC)'.

Meira-Góes, R., H. Marchand and S. Lafortune (2021*a*). 'Dealing with sensor and actuator deception attacks in supervisory control'. `https://figshare.com/s/278e1e15499ff774a75d`; *the paper will be posted on arxiv when the review process is completed.*

Meira-Góes, R., J. Weitze and S. Lafortune (2021*b*). 'A compact and uniform approach for synthesizing state-based property-enforcing supervisors for discrete-event systems'. *IEEE Transactions on Automatic Control*.

Meira-Góes, R., R. H. Kwong and S. Lafortune (2021*c*). 'Synthesis of optimal multi-objective attack strategies for controlled systems modeled by probabilistic automata'. *IEEE Transactions on Automatic Control*.

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

Ramadge, P. J. and W. M. Wonham (1987). 'Supervisory control of a class of discrete event processes'. *SIAM J. Control Optim.* **25**(1), 206–230.

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

Takai, S. (2000). 'Supervisory control of partially observed discrete event systems with arbitrary control patterns'. *International Journal of Systems Science* **31**(5), 649–656.

Teixeira, A., D. Pérez, H. Sandberg and K. H. Johansson (2012). Attack models and scenarios for networked control systems. In 'Proceedings of the 1st International Conference on High Confidence Networked Systems'. HiCoNS '12. ACM. New York, NY, USA. pp. 55–64.

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

Wakaiki, M., P. Tabuada and J. P. Hespanha (2018). 'Supervisory control of discrete-event systems under attacks'. *Dynamic Games and Applications* **9**, 965 – 983.

Wang, Y. and M. Pajic (2019). Attack-resilient supervisory control with intermittently secure communication. In '2019 IEEE 58th Conference on Decision and Control (CDC)'. pp. 2015–2020.

Wonham, W. M. and K. Cai (2018). *Supervisory Control of Discrete-Event Systems*. Springer International Publishing.

Yin, X. and S. Lafortune (2016). 'A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems'. *IEEE Transactions on Automatic Control* **61**(8), 2140–2154.

Zhang, Q., Z. Li, C. Seatzu and A. Giua (2018). Stealthy attacks for partially-observed discrete event systems. In '2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)'. Vol. 1. pp. 1161–1164.

Zhu, Y., L. Lin and R. Su (2019). Supervisor obfuscation against actuator enablement attack. In '2019 18th European Control Conference (ECC)'. pp. 1760–1765.