Towards resilient supervisors against sensor deception attacks

Rômulo Meira-Góes, Hervé Marchand and Stéphane Lafortune

***Abstract*— We consider feedback control systems where sensor readings may be compromised by a malicious attacker intent on causing damage to the system. We study this problem at the supervisory layer of the control system, using discrete event systems techniques. We assume that the attacker can *edit* the outputs from the sensors of the system before they reach the supervisory controller. 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. We solve this problem using a new methodology that improves upon prior work on this topic. The solution methodology is based on the solution of a partially observed supervisory control problem with arbitrary control patterns. We also provide results on the existence of a supremal robust supervisor.**

I. Introduction

Security concerns are a subject of increasing attention in the control community, e.g., see [1], [15]. It was only recently that security aspects started to be incorporated into the design of feedback control systems. Hence, 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 the observable events controls the behavior of the plant via actuator commands. Based on this event-driven model, we incorporate a malicious attacker that hijacks a subset of the observable events and feeds the supervisor with incorrect information about the plant; this type of attack is known as *sensor deception attack* [10]. The goal of the attacker is to leverage its knowledge of the plant model and of the control actions of the supervisor and to use its event-editing capabilities to steer the plant state to a *critical* state where damage to the plant would occur. In contrast to the objective of the attacker, the goal of the supervisor is to prevent any attacker with event-editing capabilities to cause damage to the plant. We study the design of feedback control systems that are robust against *sensor deception attacks* at the supervisory layer of a feedback control system.

Prior work on sensor deception attacks in the field of Discrete Event Systems (DES) mainly focuses on designing attack strategies for *fixed supervisors* [9], [13], [10], [11] or focuses on designing intrusion detection modules for *fixed supervisors* [2], [7]. Exceptions to that are the works in [16], [13]. It is also relevant to mention the work in actuation deception attacks in [8], [18], even though a different deception attack class is investigated in these works.

The recent works of [16], [13] studied synthesis of robust supervisors against deception attacks. In [16], the authors provide necessary and sufficient conditions for the existence of a supervisor that exactly achieves a specification under sensor deception attacks. On the other hand, the results in [13] shows how to synthesize for a given specification the supervisor that generates the supremal controllable and normal sublanguage robust against bounded sensor deception attacks. The methodology in [13] has triple exponential complexity, in the worst case, in the size of the system model. In this paper, we provide a computationally more efficient method, namely single exponential, to solve the problem of synthesis of a robust supervisor against unbounded sensor deception attacks. Another difference between [13] and this paper is the way the (observable) events are processed by the supervisor when a string is inserted: one at time in this paper but the string as a whole in [13].

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, to capture the interaction of the attacker under the constraints of the plant model. The attacked plant can be built in a manner that accounts for all possible attacks or can be based on a known attack model. The attacked plant captures at the same time the execution of events in the original plant and the information received by the supervisor. The second step poses a supervisory control problem for the attacked plant under the specification that states that cause damage to the plant should never be reached. Specifically, 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 [4], [6], [14]. We show that the solution of the supervisory control problem for the attacked plant provides a solution for the problem addressed in this paper.

The paper is organized as follows. Section II introduces necessary background used throughout the paper. The framework of supervisory control theory under sensor deception attacks and the synthesis of robust supervisors problem are presented in Section III. In Section IV, we provide the solution methodology for the studied problem and discuss its correctness. We conclude the paper in Section V. Proofs are omitted due to space limitations.

R.M.G. and S.L. are with the Department of EECS, University of Michigan, USA. Their work was supported by US NSF grant CNS-1738103. H. M. is with INRIA, Centre Rennes - Bretagne Atlantique, 35042, France (e-mail:herve.marchand@inria.fr).

II. Preliminaries

We a consider a discrete transition system modeled as a finite-state automaton. A finite-state 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 $X \times \Sigma^*$. The language generated by G is defined as $\mathcal{L}(G) = \{s \in \Sigma^* | \delta(x_0, s)!\}$, where ! means "is defined". For $x \in X_G$, we define $\Gamma_G(x) = \{e \in \Sigma \mid \delta_G(x, e)!\}$ as the active event set at state x. For $K \subseteq \Sigma^*$, we denote by $pr(K)$ as 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 [12], we consider an uncontrolled system (plant) G that needs to be controlled in order to satisfy given safety specifications. In order to control G, the event set Σ is partitioned into two disjoint sets, which are the set of controllable events Σ_c and the set of uncontrollable events Σ_{uc}. The safety specifications on G are enforced by a supervisor, denoted by S_P, that dynamically enables/disables controllable events. The resulting controlled behavior is a new DES denoted by S_P/G, resulting in the closed-loop language $\mathcal{L}(S_P/G)$, defined in the usual manner (see, e.g., [3], [17]).

In addition, when the system is partially observed due to the limited sensing capabilities of G, the event set is also partitioned into $\Sigma = \Sigma_o \cup \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. Based on this second partition, the *projection* function $P_o : \Sigma^* \to \Sigma_o^*$ is recursively defined as: $P_o(\epsilon) = \epsilon$, $P(se) = P(s)e$ if $e \in \Sigma_o$, and $P(se) = P(s)$ otherwise. The inverse projection $P_o^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ is defined as $P_o^{-1}(t) = \{s \in \Sigma^* | P(s) = t\}$.

Supervisor S_P makes its control decisions based on a string of observable events. Formally, a partially observation supervisor is a (partial) function $S_P : P_o(\mathcal{L}(G)) \to \Gamma$, where $\Gamma = \{\gamma \in 2^{\Sigma} : \Sigma_{uc} \subseteq \gamma\}$.

In this paper, we need to leverage the partially observed supervisory control problem with arbitrary control patterns (SCP-AP) studied in [4], [6], [14]. In this problem, the supervisor S_P is defined based on an arbitrary set $C \subseteq \Gamma$, i.e., $S_P : P_o(\mathcal{L}(G)) \to C$. Now, we recall a result from [14].

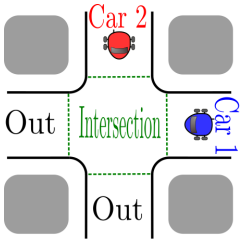
Proposition II.1. [14] Let $K \subseteq \mathcal{L}(G)$ be a non-empty and prefix-closed language and let $C \subseteq \Gamma$ be the available control patterns set. There exists a supervisor $S_P : P_o(\mathcal{L}(G)) \to C$ such that $\mathcal{L}(S_P/G) = K$ if and only if K satisfies the following condition: for any $s \in K$ and $t \in P_o^{-1}[P_o(s)] \cap \mathcal{L}(G)$, there exists a control pattern $\gamma \in C$ such that $\gamma \cap \Sigma_{\mathcal{L}(G)}(t) = \Sigma_K(t)$, where $\Sigma_K(s) = \{e \in \Sigma \mid se \in K\}$ is the one event continuation of string $s \in K$.

Proposition II.1 reduces to the standard controllability and observability conditions when $C = \Gamma$ [14]. When K does not satisfy Proposition II.1, the set $\mathcal{CO}(K) = \{K' \subseteq \mathcal{L}(G) \mid K' = pr(K') \subseteq K$ s.t. K' satisfies Proposition II.1$\}$ is defined. Note that, $\mathcal{CO}(K)$ is non-empty since $\emptyset \in \mathcal{CO}(K)$. Similarly to the standard partially observed supervisory control problem, there does not exist in general a supremal element in $\mathcal{CO}(K)$.

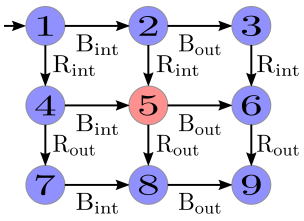
Without loss of generality, we assume that S_P is realized by an automaton $R = (X_R, \Sigma, \delta_R, x_{0,R})$ (see, e.g., [3], [17]). While the domain of events in R is Σ not Σ_o, its transitions are only driven by *observable* events. We use both notations S_P and R interchangeably.

We complete this section by defining useful notation for strings and with an example of supervisory control. For any string $s \in \Sigma^*$, $|s|$ denotes the length of s. We define by e_s^i the i^{th} event of s such that $s = e_s^1 e_s^2 \dots e_s^{|s|}$. Finally, s^i is i^{th} substring of s, namely $s^i = e_s^1 \dots e_s^i$ and $s^0 = \epsilon$.

Example II.1. We use the collision avoidance problem of vehicles at an intersection as running example throughout the paper. We have two roads with one car in each road approaching an intersection as in Figure 1(a). 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 Figure 1(b). Every event is controllable and observable and $C = \Gamma$. The supervisor realization R that guarantees the specification described can be obtained by deleting state 5.



(a) Example intersection with two cars



(b) Model of an intersection with two cars. Events with B(lue) are related to car 1, while events with R(ed) are related to car 2.

Fig. 1: Intersection example

III. Robust Supervisory Control against Sensor Deception Attacks

The left diagram of Figure 2 pictorially describes sensor deception attacks in the supervisory control framework, where the attacker intervenes in the communication channel between the plant's sensors 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 the sensor readings in this communication channel, where "alter" means that it can insert or delete events. The subset of affected sensor readings is defined as the *compromised event set* and denoted by $\Sigma_a \subseteq \Sigma_o$.

We study the problem of *synthesis of a supervisor* that guarantees a safe controlled behavior even in the presence of an attacker with the previously described deceptive capabilities. In other words, the goal is to synthesize a supervisor that

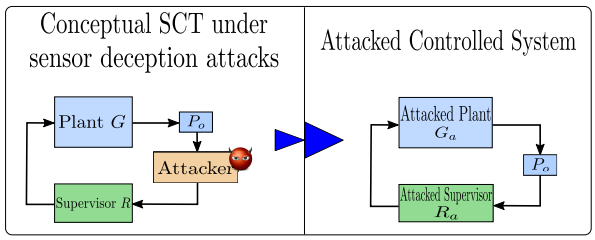


Fig. 2: Deception Attack Framework

is provably robust against deception attacks over the sensors Σ_a.

In this work, we use an approach similar to that in [2], [7]. The conceptual diagram of sensor deception attacks, as shown in Figure 2, is transformed into an attacked controlled system, which is constructed based on the original controlled system and attacker assumptions. In this attacked controlled system, we perform our analysis and provide guarantees about robustness against sensor deception attacks. In the next sections, we go over the attacker assumptions, the description of the attacked closed-loop behavior, and formally pose the problem we study.

A. Attack model

As was previously mentioned, the left diagram of Figure 2 pictorially describes sensor deception attacks in the supervisory control framework. The attacker hijacks the communication channel between the plant and the supervisor. Intuitively, the attacker receives some *information* from G and based on its *capabilities* it applies some *strategy* to attain its goal. To formally define an attacker, we must consider which information it receives, what are its capabilities and what is its strategy.

First, we assume that the attacker observes the same observable events as the supervisor. Moreover, we assume that the attacker has full knowledge of the plant G and supervisor R. This assumption is common in security analysis of systems. In our case, it means that we are studying the worst-case scenario.

Second, the attacker can only modify the compromised event set $\Sigma_a \subseteq \Sigma_o$ in order to accomplish its goal. By modify, we mean that it can insert or delete events in Σ_a over the communication channel between the plant and supervisor.

Finally, the last attacker assumption we make is related to its strategy. First, we present a general attack strategy that it is suitable whenever we do not have prior knowledge, besides Σ_a, about the attacker. The *"all-out"* attack strategy was introduced in [2], [7]. In this model, the attacker attacks whenever it is possible. 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 sensor deception attack.

On the other hand, if there exists information about the strategy that an attacker follows, then we construct a supervisor that is robust against this strategy (see Section III-C). Note that, we still consider a deception attack strategy based on Σ_a.

The next section discusses the closed-loop behavior of the attacked system based on the "all-out" strategy. Next, we discuss prior attack knowledge and how it affects the closed-loop behavior of the attacked system.

B. Closed-loop system under deception attack

Based on the previously described attacker assumptions, we define the behavior of the closed-loop attacked system. For that, we first recall the model of the attacked plant G_a. The attacked plant is defined differently than the one in [7].

We use the subscripts i and d to distinguish legitimate events generated by G from the modified events generated by the attacker, insertion and deletion respectively. We define $\Sigma_a^e = \Sigma_a^i \cup \Sigma_a^d$, where $\Sigma_a^i = \{e_i \mid e \in \Sigma_a\}$ and $\Sigma_a^d = \{e_d \mid e \in \Sigma_a\}$, to be the compromised events set with the subscripts i and d.

In the attacked plant, the events e_i simulate the insertion ability of the attacker. The insertion events e_i are self-loops in G_a since they are fictitious events, i.e., they do not change the state of the plant G. Every state of G is enhanced with insertion events.

On the other hand, the events e_d simulate the deletion ability of the attacker. These events are defined in parallel to transitions with events in Σ_a since an event must be executed by G in order to delete this information from the communication channel.

We are now ready to define the behavior of a plant G under an attack.

Definition III.1. Given G and Σ_a, we define the attacked plant G_a as: $G_a = (X_{G_a}, \Sigma_m, \delta_{G_a}, x_{0,G_a})$

1: $X_{G_a} = X_G$
2: $\Sigma_m = \Sigma \cup \Sigma_a^e$
3: $\delta_{G_a}(x,e) = \begin{cases} \delta_G(x,e) & \text{if } e \in \Sigma \text{ and } \delta_G(x,e)! \\ x & \text{if } e \in \Sigma_a^i \\ \delta_G(x,\mathcal{M}(e)) & \text{if } e \in \Sigma_a^d \text{ and } \delta_G(x,\mathcal{M}(e))! \\ \text{undefined} & \text{otherwise} \end{cases}$
where $x \in X_{G_a}$, $e \in \Sigma_m$, and $\mathcal{M} : \Sigma_m \to \Sigma$ removes the subscripts of events in Σ_m, i.e., $\mathcal{M}(e_i) = \mathcal{M}(e_d) = e$ for $e_i, e_d \in \Sigma_a^e$ and $\mathcal{M}(e) = e$ for $e \in \Sigma$
4: $x_{0,G_a} = x_{0,G}$

Similarly to the construction of G_a, we can modify the behavior of R to reflect the modifications made by an attacker on the communication channel. We assume that R respects the controllability and observability conditions.

Definition III.2. Given R and Σ_a, we define the attacked supervisor $R_a = (X_{R_a}, \Sigma_m, \delta_{R_a}, x_{0,R_a})$ as:

1: $X_{R_a} = X_R$
2: $\Sigma_m = \Sigma \cup \Sigma_a^e$

3: $\delta_{R_a}(x,e) = \begin{cases} \delta_R(x,e) & \text{if } e \in \Sigma \text{ and } \delta_R(x,e)! \\ x & \text{if } e \in \Sigma_a^d \text{ and } \delta_R(x,\mathcal{M}(e))! \\ \delta_R(x,\mathcal{M}(e)) & \text{if } e \in \Sigma_a^i \text{ and } \delta_R(x,\mathcal{M}(e))! \\ \text{undefined} & \text{otherwise} \end{cases}$

where $x \in X_{R_a}$ and $e \in \Sigma_m$.

4: $x_{0,R_a} = x_{0,R}$

The construction of R_a is similar to the construction of G_a. However, insertion and deletion events affect the supervisor in the opposite manner as they affect the plant. The deletion events e_d are self-loops in the states of R_a where the legitimate events $e \in \Sigma_a$ are defined. If the event is deleted by the attacker, then it must have been enabled by the supervisor first. They are self-loops since the supervisor does not receive any information. On the other hand, insertion events e_i are defined in parallel to their legitimate events $e \in \Sigma_a$. Note that, the construction of R_a creates a control pattern C_a based on events $e \in \Sigma_a$. Formally, $C_a = \{\gamma \in 2^{\Sigma_m} \mid \gamma \subseteq \Sigma_{uc} \wedge (\forall e \in \gamma \cap \Sigma_a, e_i \in \gamma \wedge e_d \in \gamma)\}$.

Remark: Although R_a is a modification of R, this modification does not alter the control decisions taken by R. In other words, the plant G is still supervised by the supervisor R. R_a is defined such that it takes into account the modifications made by the attacker.

The closed-loop system under sensor deception attacks is captured by the parallel composition of $G_a||R_a$, where $||$ is the parallel composition operator as in [3]. 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 a supervised system.

The language $\mathcal{L}(R_a/G_a)$ still has events in Σ_a^e, but we are interested in analyzing the language that is executed in G. P_e^G is a projection that treats events in Σ_a^e in the following manner: $P_e^G(e) = \epsilon$, if $e \in \Sigma_a^i$, $P_e^G(e) = \mathcal{M}(e)$ otherwise. Since insertion events are not executed by G, P_e^G erases them. Deletion events are mapped to their legitimate events (Σ), and legitimate events are unaltered. In this manner, the closed-loop language executed by G is defined by $P_e^G(\mathcal{L}(R_a/G_a))$, where the projection operation is extended to languages in the usual manner.

C. Known attack strategy

In the previous section, we defined the attacked behavior based on the "all-out" attack strategy. In this section, we relax this assumption based on some prior knowledge of the attack strategy used by an attacker. Using this 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. In this work, we assume that the known attack strategy is encoded as an automaton $A = (X_A, \delta_A, \Sigma_m, x_{0,A})$ as in [9].

If the automaton A encodes the attack strategy correctly, then we can construct G_a as a composition of A and G, i.e., $G_a = A||G$. In this manner, the controlled behavior of the attacked system is defined by $G_a = A||G$ and R_a as in Definition III.2.

We give one example of an attack strategy to demonstrate how G_a is constructed. We assume that the attacker can only insert events (deletion events are not allowed) and it inserts at most one event after an event is executed in G. The automaton in Figure 3 encodes this attack strategy. Composing A with G provides the attacked plant G_a.

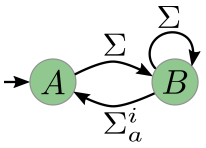


Fig. 3: Attack strategy with bounded insertions

For the rest of the paper, we only discuss results for G_a as defined in Definition III.1. However, these results are also applicable to G_a constructed based on a known attack strategy A.

D. Supervisor robust against sensor deception attacks

In this paper, we consider the *dual* problem of the work in [9]; the problem of synthesizing a supervisor R such that no attack causes damage to the plant. Therefore, we assume that the plant G contains a set of *critical* states defined as $X_{crit} \subset X$; these states are unsafe in the sense that they are states where physical 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. The attacker *succeeds*, if there exists $s \in P_e^G(\mathcal{L}(R_a/G_a))$ such that $\delta_G(x_0, s) \in X_{crit}$. In relation to the supervisor, it succeeds if X_{crit} is not reachable in the attacked system R_a/G_a.

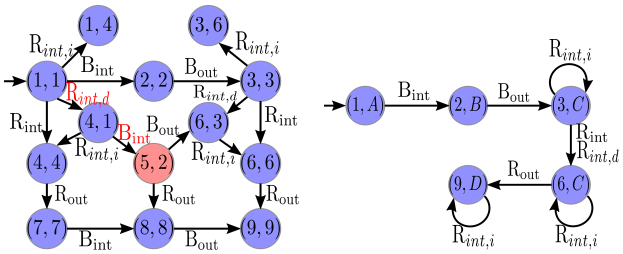
Definition III.3. Given G with X_{crit} and R, we say that R_a/G_a is safe if for any $s \in P_e^G(\mathcal{L}(R_a/G_a))$, then $\delta_G(x_0, s) \notin X_{crit}$. In this case, R is *robust against sensor deception attacks over* Σ_a, or simply robust.

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

Problem III.1 (RSS-Robust Supervisor Synthesis)**.** Given G with X_{crit} and $\Sigma_a \subseteq \Sigma_o$, synthesize a robust supervisor R, if one exists, such that R_a/G_a is safe.

Example III.1. Let us return to our intersection example. Let us assume that $\Sigma_a = \{R_{int}\}$. Using G and R as given in Example II.1, we construct the pair G_a and R_a. Thus, we can analyze the system R_a/G_a. The attacked system R_a/G_a is shown in Figure 4(a). The string $s = R_{int,d}B_{int}$, in red in Figure 4(a), is feasible in R_a/G_a, then $P_e^G(s) = R_{int}B_{int}$ is executed in G, which takes it to state 5.

Figure 4(b) demonstrates the attacked behavior for a supervisor R', different than R, for G. Note that, R' is robust since there does not exist any string that reaches the critical state. In the next section, we show how to compute this R'.



(a) The attacked controlled system R_a/G_a

(b) The attacked controlled system for R'_a/G_a

Fig. 4: Closed-loop behavior of the attacked system

IV. Solution to the Robust Supervisory Control Problem

Example III.1 shows that the supervisor designed using the usual supervisory control theory fails to provide robustness against deception attacks. It fails since the usual methodology does not take into account sensor deception attacks. Our goal is to enhance the supervisory control framework such that we answer the **RSS** problem.

Inspired by the technique used in [2], the model G_a defined in Section III becomes the system at the center of our study. Since the attacked plant G_a captures the behavior of plant G under a sensor deception attack over Σ_a, our solution technique poses a supervisory control problem directly at the attacked plant G_a. Although it appears that we could directly apply standard supervisory control techniques, the events in Σ_a^e prevent us to do so because we cannot assign them to the controllable/uncontrollable and the observable/unobservable partitions. Next, we provide a solution to this problem starting with the controllability issue.

The controllability of events in Σ_a^e is directly related to their legitimate counterparts. These events cannot be directly controlled since an attacker decides when to attack. However, if a compromised event $e \in \Sigma_a$ is controllable, then the supervisor can control when to disable e, which consequently disables events e_i and e_d. The latter is a consequence of the assumption that the attacker does not insert events that are disabled at a current control decision (see Definition III.2). Therefore, events in Σ_a^e depend on the decision made on their legitimate counterparts. As mentioned in Definition III.2, this dependence provides us *the control patterns* C_a that the supervisor for G_a must follow.

The control patterns created by Σ_a^e solve the issue related to controllability of events in Σ_a^e but they do not solve the issue related to observability. P_e^G projects attacked strings from G_a to strings executed in G, but it does not capture how attacked strings are perceived by the supervisor. Even though the system R_a/G_a captures exactly the closed-loop behavior of the attacked system, we are interested in synthesizing a supervisor R that has Σ_o^* as input. In other words, we use R_a/G_a to test if a supervisor R with Σ_o^* as input is safe. Thus, we define $P_e^R : \Sigma_m \to \Sigma_o$ as: $P_e^R(e) = \epsilon$, if $e \in \Sigma_a^d$; and $P_e^R(e) = \mathcal{M}(e)$ otherwise.

Let us analyze G_a with the projection P_e^R. Recall that Definition III.1 adds self-loops of insertions events at each state of G. It also adds transitions with deletion events parallel to transitions with their legitimate event. Therefore, if we use the projection P_e^R in G_a, then the insertion events are mapped to their legitimate events and deletion events are mapped to the empty string. This operation generates a nondeterministic automaton, which we want to avoid. We prefer a partially observed system to be consistent with standard supervisory control.

We consider the following procedure to eliminate the nondeterminism generated by P_e^R and analyze strings from G_a as they are seen by the supervisor.

Definition IV.4. Given G_a, we define $G_m = (X_{G_m}, \Sigma_m, \delta_{G_m}, x_{0,G_m})$ as:

1: $X_{G_m} = X_{G_a} \cup (X_{G_a} \times \{\sigma_i \mid \sigma \in \Sigma_a\})$

2: Σ_m

3: $\delta_{G_m}(x,e) = \begin{cases} \delta_{G_a}(x,e) & \text{if } x \in X_{G_a} \text{ and } e \notin \Sigma_a^i \\ (x,e) & \text{if } x \in X_{G_a} \text{ and } e \in \Sigma_a^i \\ x_1 & \text{if } x \in X_{G_a} \times \{\sigma_i | \sigma \in \Sigma_a\}, \; x = (x_1, e_1), \; e = \mathcal{M}(e_1) \\ \text{undefined} & \text{otherwise} \end{cases}$

4: $x_{0,G_m} = x_{0,G_a}$

Given the automaton G_m, we can know discuss the observable events of this system. After the execution of an event e_i, the legitimate counterpart e is executed in G_m. The execution of an event e_d is still considered unobservable in G_m. Therefore, we can now specify the events in Σ_a^e as unobservable. Moreover, the events that are unobservable in G continue to be unobservable in G_m. Thus, the unobservable event set for G_m is $\Sigma_{m,uo} = \Sigma_{uo} \cup \Sigma_a^e$ and the observable set is $\Sigma_{m,o} = \Sigma_o$. We also define $P_{m,o}$ as the projection operation of strings in Σ_m to $\Sigma_{m,o}$. The following proposition shows that the language observed by a supervisor R through G_a is the same as the one observed by G_m.

Proposition IV.2. $P_o(P_e^R(\mathcal{L}(G_a))) = P_{m,o}(\mathcal{L}(G_m))$

The attacked system G_m has $\Sigma_{m,o} = \Sigma_o$ as the set of observable events in G_m and C_a as the set of control patterns. To be able to use the results from [14], we need the following proposition.

Proposition IV.3. The set C_a is closed under union.

Since C_a is closed under union, the results in [14] are applicable to G_m. Let $K = \mathcal{L}(trim(G_m, X_{crit}))$ be the language specification on $\mathcal{L}(G_m)$, where $trim(G_m, X_{crit})$ is the *accessible* subautomaton of G_m after deleting states $X_{crit} \subseteq X_{G_m}$. From [14], it follows that for any $K' \in \mathcal{CO}(K)$, if $K' \neq \emptyset$, there exist a supervisor $S_P : P_o(\mathcal{L}(G_m)) \to C_a$ such that $\mathcal{L}(S_P/G_m) = K'$. Also, the supremal element of $\mathcal{CO}(K)$ does not exist in general. For this reason, we search for a supervisor that generates a maximal K'.

Problem IV.2 (**SCP-AP**-Supervisory Control with Arbitrary Control Patterns)**.** Given the attacked system G_m, and

$K = \mathcal{L}(trim(G_m, X_{crit}))$. Let $K' \in \mathcal{CO}(K)$, $K' \neq \emptyset$, such that $\nexists L \in \mathcal{CO}(K)$ with $K' \subseteq L$, namely K' is a maximal sublanguage in $\mathcal{CO}(K)$. Synthesize a supervisor $S_P : P_o(\mathcal{L}(G_m)) \to C_a$ that satisfies $\mathcal{L}(S_P/G_m) = K'$.

Since the languages $\mathcal{L}(G_m)$ and K are regular languages, the supervisor S_P can be encoded by a DFA R_m. Next, we define a supervisor R for the **RSS** problem based on the supervisor R_m. Using this supervisor, we prove that the **RSS** problem is reducible to the **SCP-AP** problem.

Definition IV.5. Assume that $\mathcal{CO}(K) \neq \emptyset$ in the **SCP-AP** problem. Let the supervisor R_m be a solution for the **SCP-AP** problem. Construct the supervisor $R = (X_R, \Sigma, \delta_R, x_{0,R})$ in the following manner.

1: $X_R = X_{R_m}$
2: $\Sigma = \Sigma$
3: $\delta_R(x,e) = \begin{cases} \delta_{R_m}(x,e) & \text{if } x \in X_R \text{ and } e \in \Sigma \\ \text{undefined} & \text{otherwise} \end{cases}$
4: $x_{0,R} = x_{0,R_m}$

Definition IV.5 constructs the supervisor R by simply removing the events in Σ_a^e. The transitions with events in Σ_a^e on R_m are self-loops since they are unobservable on the **SCP-AP** problem. We now provide the connection between the **RSS** problem and the **SCP-AP** problem.

Theorem IV.1. The **RSS** problem is reducible to the **SCP-AP** problem.

Remark 1: Theorem IV.1 provides a reduction of the **RSS** problem to the partially observed supervisory control problem with arbitrary control patterns. Although an algorithm to synthesize a supervisor was not provided in [14], one can adapt standard algorithms that compute maximal controllable and observable sublanguages to consider control patterns. For example, the VLP-PO algorithm from [5] can be adapted such that it considers control patterns. Based on the runtime of the original VLP-PO algorithm, the **RSS** problem can be solved in $O\big((|\Sigma_c|^2(|X_G|+|X_G||\Sigma_a|))e^{|X_G|+|X_G||\Sigma_a|}\big)$ time.

Remark 2: Event in the case of G being fully observable, the supremal robust language element does not exist in general. Sufficient and necessary conditions for the existence of the supremal robust supervisor exist. A necessary but not sufficient condition is the familiar normality condition ($\Sigma_c \subseteq \Sigma_o$) [3], [17]. A second condition is needed, i.e., $\Sigma_a \subseteq \Sigma_{uc}$.

V. Conclusion

We have considered a class of problems in cyber-security where sensor readings in a feedback control system may be manipulated by a malicious attacker. We leverage techniques from supervisory control with arbitrary control patterns of partially-observed discrete event systems to develop a solution methodology to prevent damage to the system when some sensor readings may be edited by the attacker. Our problem formulation considered that the attacker perfectly knows the plant and the synthesized supervisor and there is not a detection module explicitly trying to diagnose sensor attacks. Thus, the supervisor alone must ensure robustness against sensor deception attacks. We showed how this problem can be reduced to a partially observed supervisory control problem with arbitrary control patterns. Our solution methodology has single exponential complexity over the number of states of the plant and events. It is more computationally efficient than the previous method in the literature [13].

It would be of interest to merge the techniques of robust synthesis with intrusion detection modules. Moreover, it would be interesting to relax some assumptions on the attacker. For example, introducing costs on the attacker modifications might be interesting.

References

[1] A. A. Cardenas, S. Amin, and S. Sastry. Secure control: Towards survivable cyber-physical systems. In *2008 The 28th International Conference on Distributed Computing Systems Workshops*, pages 495–500, June 2008.

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

[3] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2008.

[4] C. H. Golaszewski and P. J. Ramadge. Control of discrete event processes with forced events. In *26th IEEE Conference on Decision and Control*, volume 26, pages 247–251, Dec 1987.

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

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

[7] P. M. Lima, L. K. Carvalho, and M. V. Moreira. Detectable and undetectable network attack security of cyber-physical systems. In *14th IFAC Workshop on Discrete Event Systems WODES 2018*, volume 51, pages 179 – 185, 2018.

[8] L. Lin, S. Thuijsman, Y. Zhu, S. Ware, R. Su, and M. Reniers. Synthesis of Successful Actuator Attackers on Supervisors. *ArXiv e-prints - http://arxiv.org/abs/1807.06720*, 2018.

[9] R. Meira-Góes, E. Kang, R. Kwong, and S. Lafortune. Stealthy deception attacks for cyber-physical systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4224–4230, Dec 2017.

[10] R. Meira-Góes, E. Kang, R. Kwong, and S. Lafortune. Synthesis of sensor deception attacks at the supervisory layer of cyber-physical systems. *under review at Automatica*, 2019.

[11] R. Meira-Góes, R. Kwong, and S. Lafortune. Synthesis of sensor deception attacks for systems modeled as probabilistic automata. In *2019 American Control Conference (ACC)*, July 2019.

[12] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, January 1987.

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

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

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

[16] M. Wakaiki, P. Tabuada, and J. P. Hespanha. Supervisory control of discrete-event systems under attacks. *Dynamic Games and Applications*, Sep 2018.

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

[18] Y. Zhu, L. Lin, and R. Su. Supervisor obfuscation against actuator enablement attack. *ArXiv e-prints - http://arxiv.org/abs/1811.02932*, 2018.