2017 IEEE 56th Annual Conference on Decision and Control (CDC)
December 12-15, 2017, Melbourne, Australia

Stealthy Deception Attacks for Cyber-Physical Systems

Rômulo Meira Góes, Eunsuk Kang, Raymond Kwong and Stéphane Lafortune

Abstract— **We study the security of Cyber-Physical Systems (CPS) in the context of the supervisory control layer. Specifically, we propose a general model of a CPS attacker in the framework of Discrete Event Systems (DES) and investigate the problem of synthesizing an attack strategy for a given controlled system. Our model captures a class of *deception attacks*, where the attacker has the ability to modify a subset of sensor readings and mislead the supervisor, with the goal of inducing the system into an undesirable state. We introduce a new type of a bipartite transition structure, called *Insertion-Deletion Attack structure* (IDA), to capture the game-like interaction between the supervisor and the environment (which includes the system and attacker). This structure is a discrete transition system that embeds information about all possible attacker's stealthy actions, and all states (some possibly unsafe) that become reachable as a result of those actions. We present a procedure for the construction of the IDA and discuss its properties. Based on the IDA, we discuss the characterization of successful stealthy attacks, i.e., attacks that avoid detection from the supervisor and cause damage to the system.**

I. Introduction

In this paper, we are concerned with the problem of synthesizing an attack strategy at the supervisory control layer of a given Cyber-Physical Systems (CPS). Previously, some efforts were made in classification and modeling of cyber-attacks, assuming certain intelligence on the part of the attacker; see, e.g., [1], [2]. Our focus is a special type of attacks, called *deception attacks*, which are characterized by some type of manipulation of the sensor measurements received by the controller/supervisor. Given that we are investigating cyber-attacks at the supervisory control layer of a CPS, we use the formalism of Discrete Event Systems (DES) to model both the attacker's behavior as well as the CPS behavior itself. This allows us to leverage the concepts and techniques of the theory of supervisory control of DES. Several recent works have adopted similar approaches to study cyber-security issues in CPS; see, e.g., [3], [4], [5], [6].

Previous works [3], [4], [5] on intrusion detection and prevention of cyber-attacks using discrete event models were focused on modeling the attacker as faulty behavior and their corresponding methodologies were relying on fault diagnosis techniques. Recently, [7] proposed a framework similar to the one adopted in our paper, where they formulate a model of data deception attacks. However, our methodology is more general than that in [7], as it allows arbitrary insertions or deletions of events. Furthermore, in [7], a normality condition is necessary to create the attack strategy; this condition is imposed to obtain the so-called supremal controllable and normal language under the attack model. In our approach, this condition is relaxed, and normality is not necessary to create an attack strategy, thus allowing a larger class of attacks strategies. In [6], the authors presented a study of supervisory control of DES under attacks. They introduced a new notion of observability which captures the presence of an attacker. However, their study is focused on the supervisor's viewpoint and they do not develop a methodology to design attack strategies. They assume that the attacker's model is given and they develop their results based on that assumption. In that sense, the work [6] is closer to robust supervisory control and it is complementary to our work. Several prior works considered robust supervisory control under different notions of robustness [8], [9], [10], [11], but they did not study robustness against attacks. In the cyber-security literature, some works have been carried out in the context of discrete event models, especially regarding opacity and privacy/secrecy properties [12], [13], [14], [15]. These works are concerned with studying information release properties by the system, and they do not address the impact of an intruder over the physical parts of the system.

In this paper, we propose a model of deception attacks at the supervisory control layer and introduce the general problem of synthesis of *successful stealthy deception attacks*. We assume an attacker with (i) knowledge of both the system and its supervisor and (ii) ability to affect the sensor information that is received by the supervisor. The goal of the attacker is to induce the supervisor into allowing the system to reach an unsafe state, thereby causing damage to the system. The set of unsafe states is assumed to be pre-specified. The methodology that we develop for investigating the synthesis of such attacks is inspired by the work in [16], [17], [15]; as in these works, we employ a bipartite discrete structure to model the game-like interaction between the supervisor and the environment (system and attacker). We call our new structure the *All Stealthy Insertion-Deletion Attack* structure (or IDA). By construction, the IDA embeds all possible scenarios where the attacker inserts or deletes some subset of observable events without being noticed by the supervisor. The IDA, once constructed, serves as the basis for solving the synthesis problem.

By providing a general analysis and synthesis framework, our goal is to allow CPS engineers to detect and

This work was supported in part by the U.S. National Science Foundation grant CNS-1421122.

Rômulo Meira Góes, Eunsuk Kang, and Stéphane Lafortune are with Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48105, USA {romulo,eskang,stephane}@umich.edu

Raymond Kwong is with the Department of Electrical Engineering and Computer Science, University of Toronto, Toronto, ON M5S 3G4, Canada kwong@control.utoronto.ca

address potential vulnerabilities in their controlled systems. To demonstrate our approach, we performed a case study on the security of a realistic, fully operational water treatment testbed investigated in prior related research [18].

The remainder of this paper is organized as follow. Section II introduces necessary background and some notations. Section III formalizes the attack model as well as the problem statement. Section IV describes the IDA structure and its properties. Section V briefly describes results in the case study. Lastly, Section VI provides conclusions and directions for future work.

II. Modeling Formalism

We use the formalism of DES modeled as finite state automata. A Finite-State Automaton G is defined as a tuple $G = (X, \Sigma, \delta, x_0)$, where

- X is a finite set of states;
- Σ is a finite set of events;
- $\delta : X \times \Sigma \to X$ is a partial transition function;
- $x_0 \in X$ is the initial state.

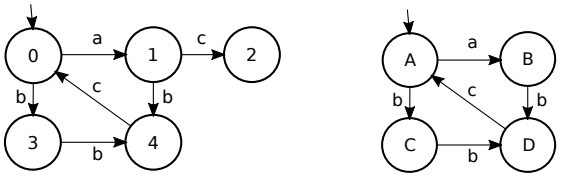
The function δ is extended in the usual manner to domain $X \times \Sigma^*$. The language generated by G is defined as $\mathscr{L}(G) = \{s \in \Sigma^* | \delta(x_0, s)!\}$, where ! means "is defined". Language $\mathscr{L}(G)$ is considered as the uncontrolled system behavior, since it includes all possible executions of G. We assume that a supervisor S_P was designed to enforce some safety and/or nonblockingness (or liveness) property on G. In the notation of the theory of supervisory control of DES initiated in [19], the resulting controlled behavior is a new DES denoted by S_P/G, resulting in the closed-loop language $\mathscr{L}(S_P/G)$, defined in the usual manner [20]. S_P dynamically enables and disables the controllable events of G (i.e., the actuators), on the basis of the observable events of G that it tracks (from the sensors of G). The limited actuation capabilities of G are modeled by a partition in the event set $\Sigma = \Sigma_c \cup \Sigma_{uc}$, where Σ_{uc} is the set of uncontrollable events and Σ_c is the set of controllable events. The set of admissible control decisions is defined as $\Gamma = \{\gamma \in 2^{\Sigma} : \Sigma_{uc} \subseteq \gamma\}$, where admissibility guarantees that a control decision will never disable an uncontrollable event. 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 $P : \Sigma^* \to \Sigma_o^*$ is defined as :

$$P(\varepsilon) = \varepsilon \text{ and } P(se) = \begin{cases} P(s)e & \text{if } e \in \Sigma_o \\ P(s) & \text{if } e \in \Sigma_{uo} \end{cases} \quad (1)$$

The inverse projection $P^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ is defined as $P^{-1}(t) = \{s \in \Sigma^* | P(s) = t\}$. In addition, $\Gamma_G(S)$ is defined as the set of active events at the subset of states $S \subseteq X$ of automaton G, given by:

$$\Gamma_G(S) := \{e \in \Sigma | (\exists u \in S) \text{ s.t. } \delta(u, e)!\} \quad (2)$$

The supervisor S_P makes its decisions based on the string of observable events that it observes. Formally, a partially observation supervisor is a function $S_P : P(\mathscr{L}(G)) \to \Gamma$. Without loss of generality, we assume that S_P is realized as a deterministic automaton $R = (Q, \Sigma, \mu, q_0)$ such that $\forall q \in Q$, if $e \in \Sigma_{uo}$ is an enabled unobservable event at state q by S_P, then we define $\mu(q, e) = q$ as is customary in a supervisor realization (cf. [20]). This means that the supervisor can only change its control decision (by updating the state in R) upon the occurrence of an observable event; yet, its active event set is the actual control decision issued to the system, including the enabled unobservable events. More explicitly, the current control decision applied to G is $\Gamma_R(\{q\})$, where Γ_R is set of active events at state q of R. Thus, while the domain of events in R is Σ not Σ_o, its transitions will only be driven by the strings of *observable* events that it receives from the attacker.



(a) G (b) Supervisor R_1

Fig. 1: System automaton along with its supervisor

We also define $\tilde{R} = (\tilde{Q}, \Sigma, \tilde{\mu}, q_0)$, where $\tilde{Q} = Q \cup \{\text{dead}\}$, and $\tilde{\mu}$ is defined by completing the partial transition function μ as $(\forall q \in Q)\ (\forall e \in [\Sigma_o \backslash \Gamma_R(\{q\})])\ \tilde{\mu}(q, e) =$ dead. The state "dead" will allow us to capture when the controlled system goes "out of range" of the closed-loop behavior for which S_P was designed; at the same time, this state will capture when the supervisor detects that it is under attack.

Example II.1. Consider the system G represented in Fig. 1(a). Let $\Sigma = \Sigma_o = \{a, b, c\}$ and $\Sigma_c = \{b, c\}$. Figure 1(b) shows the realization R_1 of a supervisor S_{P1} that was designed for G. In this case, the language generated by $\mathscr{L}(S_{P1}/G)$ guarantees that state 2 is unreachable in the controlled behavior.

Next, let us consider that $\Sigma_o = \{a, b\}$ and $\Sigma_c = \{b, c\}$. A realization R_2 of a supervisor S_{P2} is shown in Fig. 2. The controlled behavior remains the same as the previous case.

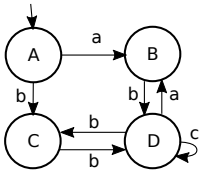


Fig. 2: Supervisor R_2

For convenience, we define two operators that will be used in this paper. The unobservable reach of the subset of states $S \subseteq X$ under the subset of events $\gamma \subseteq \Sigma$ is given by:

$$UR_{\gamma}(S) := \{x \in X | (\exists u \in S)(\exists e \in (\Sigma_{uo} \cap \gamma)^* \text{ s.t. } x = \delta(u, e)\} \quad (3)$$

The observable reach of the subset of states $S \subseteq X$ given the

execution of the observable event $e \in \Sigma_o$ is defined:

$$Next_e(S) := \{x \in X | \exists u \in S \text{ s.t. } x = \delta(u,e)\} \quad (4)$$

III. Problem Statement

In this section, we formulate the *Stealthy Insertion-Deletion Sensor Attack Problem*. Let us first define how an attacker interacts with the controlled system. Figure 3

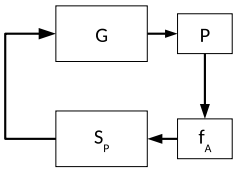


Fig. 3: Closed loop system under Deception Attack

shows a controlled system under attack, where the attacker intervenes in the communication channel between the plant's sensors and the supervisor. Specifically, the attacker has the ability to observe the same observable events as the supervisor. Moreover, we assume that the attacker possesses the ability to alter some of the sensors readings in this communication channel. By altering, we mean that it may *insert* or *delete* some subset of sensor readings sent to the supervisor; this subset is defined as the *compromised event set* $\Sigma_a \subseteq \Sigma_o$. Formally, we model an attacker with such capabilities as a string edit function.

Definition III.1. Given a system G and a subset $\Sigma_a \subseteq \Sigma_o$, an attacker is defined as a function $f_A : P(\mathscr{L}(G)) \times (\Sigma_o \cup \{\varepsilon\}) \to \Sigma_o^*$ s.t. f_A satisfies the following constraints:

- $f_A(\varepsilon,\varepsilon) \in \Sigma_a^*$;
- $\forall s \in P(\mathscr{L}(G))$, $e \in \Sigma_o \backslash \Sigma_a$: $f_A(s,e) \in \{e\}\Sigma_a^*$;
- $\forall s \in P(\mathscr{L}(G))$, $e \in \Sigma_a$: $f_A(s,e) \in \Sigma_a^*$.

The function f_A captures a general model of deception attack. Given the past output string s of G and observing a new event e, the attacker may choose to edit e if it belongs to Σ_a. The first case in the above definition gives an initial condition for an attack. The second case constrains the attacker to be unable to erase e when e is outside of Σ_a. However, the attacker may insert an arbitrary string $t \in \Sigma_a^*$ after the occurrence of e. Lastly, the third case in Definition **III.1** means event $e \in \Sigma_a$ is edited to some string $t \in \Sigma_a^*$, capturing the deletion of e as well as the insertion of t. Note that in this third case: (i) t may be equal to e, allowing the attacker to leave the event unmodified if it chooses to do so; (ii) e may be the first event of t, meaning that e is not deleted.

For convenience, let us define a string-based edit (partial) function $\hat{f}_A : P(\mathscr{L}(G)) \to \Sigma_o^*$ recursively as $\hat{f}_A(te) = \hat{f}_A(t) f_A(t,e)$, and $\hat{f}_A(\varepsilon) = f_A(\varepsilon,\varepsilon)$.

The existence of an attacker in the controlled system induces a new controlled language. More specifically, S_P and $\hat{f}_A$ together effectively generate a new supervisor S_A for system G. Formally, for any $s \in P(\mathscr{L}(G))$, $S_A(s) = [S_P \circ \hat{f}_A(s)]$, from which a new controlled language $\mathscr{L}(S_A/G)$ is constructed (in the usual manner in supervisory control theory). The composition operation $\circ$ captures the new edited observed string that now drives S_P in the presence of the attacker.

Next, let us consider the objective for an attacker. We assume that the system G contains a set of *critical* unsafe states defined as $X_{crit} \subset X$ such that $\forall x \in X_{crit}, (\forall s = e_1 ... e_n \in \mathscr{L}(G)$ s.t. $\delta(x_0,s) = x \wedge |s| = n)(\exists i \in \{1,...,n-1\})$: $e_{i+1} \notin S_P(P_o(e_1...e_i))$. In general, not all states reached by strings of G that are disabled by S_P are critically unsafe. In practice, there will be certain states among those that force the supervisor to go "out of range" that correspond to physical damage to the system, such as "overflow" states or "collision" states, for instance. Similar notions of critical unsafe states have been used in other works, e.g., [21], [4]. Therefore, the objective of the attacker is to force the controlled behavior under attack $\mathscr{L}(S_A/G)$ to reach any state in X_{crit}. At the same time, the attacker does not wish to be detected by S_P, meaning that realization $\tilde{R}$ should never enter the state "dead".

Problem III.1 (Synthesis of Stealthy Deception Attacks)**.** Given a system G, a supervisor S_P realized as an automaton $\tilde{R}$, and a set of compromised events $\Sigma_a \subseteq \Sigma_o$, synthesize an attacker f_A such that it generates a controlled language $\mathscr{L}(S_A/G)$ that satisfies:

1. $\forall s \in \mathscr{L}(S_A/G)$, $\hat{f}_A(P(s))$ is defined;
2. $\forall s \in \mathscr{L}(S_A/G)$, $\hat{f}_A(P(s)) \in P(\mathscr{L}(S_P/G))$;

3(a). $\exists s \in \mathscr{L}(S_A/G)$, s.t. $(\forall t \in [P^{-1}(P(s)) \cap \mathscr{L}(G)])$ $\delta(x_0,t) \in X_{crit}$.

In this case, we say that f_A is a *strong attacker*. We additionally define the notion of a *weak attacker* as follows:

3(b). $\exists s \in \mathscr{L}(S_A/G)$, s.t. $(\exists t \in [P^{-1}(P(s)) \cap \mathscr{L}(G)])$ $\delta(x_0,t) \in X_{crit}$.

In the formulation of the above problem, the attacker function needs to be well defined for all projected strings in the modified controlled language $P(\mathscr{L}(S_A/G))$. Condition 2 guarantees the stealthy behavior of the attacker, meaning any string in $P(\mathscr{L}(S_A/G))$ should be modified to a string within the original controlled behavior. In this manner, $\tilde{R}$ never reaches state "dead". Lastly, condition 3 exploits the reachability of critical states, where condition 3(a) is a strong version of the problem. In the strong case, the attacker is sure that the system has reached a critical state if string s occurs in the system. Condition 3(b) is a relaxed version, where the attacker is not sure if a critical state was reached, although it could have been reached. Both variations of condition 3 guarantee the existence of at least one successful attack, namely, when string s occurs in the new controlled behavior.

IV. All Stealthy Insertion-Deletion Attacks

A. Definition

The All Stealthy Insertion-Deletion Attack structure (IDA) is an extension of the *bipartite transition structure* presented in [16]. The IDA captures the game between the environment and supervisor considering the possibility that a subset of the sensor network channels may be compromised by a

malicious attacker. Consequently, the IDA must be able to capture the difference between the actual information from the system output and the information observed by the supervisor, where the latter is induced by the attacker when it provides the supervisor with false sensor events. In order to capture this difference, we define an information state *IS* as a *pair* $IS \in 2^X \times \tilde{Q}$, and the set of all information states as $I = 2^X \times \tilde{Q}$. The first element in *IS* represents the *correct* IS, i.e., the true information state of the system, as seen by the attacker for the actual system outputs. The second element represents the supervisor's state, which is the current state of its realization based on the edited string of events that it receives. Recall that Σ_a is the set of compromised events, and we call the events in $\Sigma \setminus \Sigma_a$ *uneditable*. In order to construct the IDA structure, let $\Sigma_a^i = \{e_i |\ e \in \Sigma_a\}$ be the set of inserted events and $\Sigma_a^d = \{e_d |\ e \in \Sigma_a\}$ be the set of deleted events. Note that, Σ_a^i and Σ_a^d are disjoint, and both are disjoint with Σ_a. For convenience, we also define $\Sigma_a^e = \Sigma_a^i \cup \Sigma_a^d$ to be the set of edited events. We are using these notations for clarity of the methodology; from the supervisor's point of view, $e_i = e$ and $e_d = \varepsilon$, $\forall e_i \in \Sigma_a^i$ and $\forall e_d \in \Sigma_a^d$. Furthermore, let us define the projection operation $P_e : \Sigma_a^e \to \Sigma_a$ as $P_e(e_i) = P_e(e_d) = e$.

Definition IV.2. An All Insertion-Deletion Attack structure (IDA) A w.r.t. G, Σ_a, and $\tilde{R}$, is a 7-tuple

$$A = (Q_S, Q_E, h_{SE}, h_{ES}, \Sigma, \Sigma_a^e, y_0) \tag{5}$$

where,

- $Q_S \subseteq I$ is the set of S-states, where S stands for Supervisor and $y_1 \in 2^X$ and $y_2 \in \tilde{Q}$ denote the first and second element of a S-state y, respectively. Thus, $y = (y_1, y_2)$;
- $Q_E \subseteq I \times \Gamma$ is the set of E-states, where E stands for Environment and where $I_1(z)$, $I_2(z)$, and $\Gamma(z)$ denote the correct IS, the supervisor's state, and the control decision components of a E-state z, respectively. Thus $z = (I_1(z), I_2(z), \Gamma(z))$;
- $h_{SE} : Q_S \times \Gamma \to Q_E$ is the partial transition function from S-states to E-states, satisfying the following constraint:

$$\begin{aligned} &h_{SE}(y, \gamma) := z, \text{ where} \\ &[I_1(z) = UR_\gamma(y_1)] \wedge [I_2(z) = y_2] \wedge \\ &[\Gamma(z) = \gamma = \Gamma_R(\{y_2\}) \cup \Sigma_{uc}] \end{aligned} \tag{6}$$

  (Note that h_{SE} is only defined for y and γ such that $\gamma = \Gamma_R(\{y_2\}) \cup \Sigma_{uc}$.)
- $h_{ES} : Q_E \times (\Sigma_o \cup \Sigma_a^e) \to Q_S$ is the partial transition function from E-states to S-states, satisfying the following constraints: for any $y \in Q_S$, $z \in Q_E$ and $e \in \Sigma_o \cup \Sigma_a^e$, we have:

$$h_{ES}(z, e) := y = (y_1, y_2) \tag{7}$$

$$\Rightarrow \begin{cases} [y_1 = Next_e(I_1(z)),\ y_2 = \tilde{\mu}(I_2(z), e)] \\ \quad \textbf{if } e \in \Gamma(z) \cap \Sigma_o \cap \Gamma_G(I_1(z)) & \text{(8a)} \\ [y_1 = I_1(z),\ y_2 = \tilde{\mu}(I_2(z), P_e(e))] \\ \quad \textbf{if } e \in \Sigma_a^i \text{ and } P_e(e) \in \Sigma_a \cap \Gamma_R(\{I_2(z)\}) & \text{(8b)} \\ [y_1 = Next_{P_e(e)}(I_1(z)),\ y_2 = I_2(z)] \\ \quad \textbf{if } e \in \Sigma_a^d \text{ and } P_e(e) \in \Gamma(z) \cap \Sigma_a \cap \Gamma_G(I_1(z)) & \text{(8c)} \end{cases}$$

- Σ is the set of events of G;
- Σ_a^e is the set of edited events;
- $y_0 \in Q_S$ is the initial S-state: $y_0 := (\{x_0\}, q_0)$;

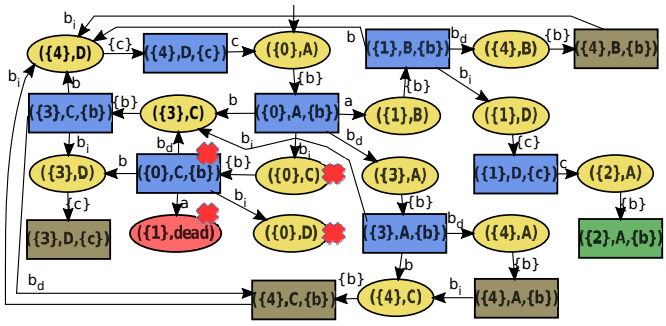
Since the purpose of the IDA is to capture the game between the supervisor and the environment, we use a bipartite structure to represent each entity. An S-state is a pair *IS* containing the correct IS and the supervisor's state; it is where the supervisor issues its control decision. An E-state is a pair *IS* and a control decision, for which the environment (system or attacker) selects one among the observable events to occur. A transition from an S-state to an E-state represents the updated unobservable reach in the correct IS together with the current supervisor state and its control decision. A transition from an E-state to an S-state represents the "observable reach" immediately following the execution of the observable event by the environment. In this case, both the correct IS and the supervisor's state are updated. However, the update of the correct IS and of the supervisor's state depends on the type of event generated by the environment: true system event unaltered by the attacker, (fictitious) event insertion by attacker, or deletion by attacker of an event just executed by the system. Thus, the transition rules are split into three cases, described below.

The partial transition function h_{ES} is characterized by three cases: Equations (8a),(8b), and (8c). Equation (8a) is related to *system* actions, while Equations (8b) and (8c) are related to *attacker* actions. In the case of Equation (8b), the attacker only inserts events consistent with the control decision of the current supervisor state (which was enhanced to include uncontrollable events). In the case of Equation (8c), it only deletes actual observable events generated by the system. In the case of Equation (8a), the system generates a feasible (enabled) event and the attacker lets the event reach the supervisor intact, either because it cannot compromise that event, or because it chooses not to make a move.

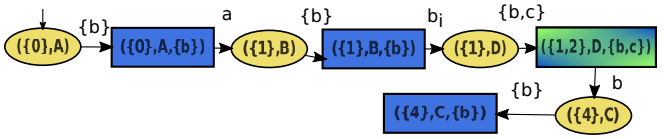
Example IV.2. As a continuation of Example **II.1**, let us construct the IDA structure for the given G and R_1, additionally with $\Sigma_a = \{b\}$ and $X_{crit} = \{2\}$. Figure 4(a) shows the resulting structure where oval states represent the S-states and rectangular states represent the E-states. Moreover, red states indicates where the supervisor reaches the "dead" state, green states represent the successful reachability of a critical states, and brown states represent system deadlocks (for simplicity, we do not construct the IDA beyond green states). In this example, the attacker is able to reach, with

certainty, a critical state. However, there exists a trace for which the attacker might be discovered, due to the occurrence of uncontrollable and uneditable event a that leads to state "dead" in the supervisor. Later in this section, we investigate the elimination of such traces by performing further processing (pruning) on the IDA.

As a second example, we consider again G but with supervisor R_2 from Example **II.1**, where $\Sigma_a = \{b\}$ and $X_{crit} = \{2\}$. The IDA in Fig. 4(b) (partially constructed due to limited space) demonstrates how a partially observed system may introduce uncertainty for the attacker. In this example, the attacker does not know for certain whether its attack succeeded. The correct information at state $(\{1,2\}, D, \{b,c\})$ contains critical as well as non-critical states, meaning the system might not end up in a critical state. We further investigate this subject later in this section.



(a) IDA for Example **IV.2** part 1



(b) Partially constructed IDA for Example **IV.2** part 2

Fig. 4: Results of Example **IV.2**

Given two IDAs A_1 and A_2, we say that A_1 is a subsystem of A_2, denoted by $A_1 \sqsubseteq A_2$, if $Q_E^{A_1} \subseteq Q_E^{A_2}$, $Q_S^{A_1} \subseteq Q_S^{A_2}$, and for any $y \in Q_S^{A_1}$, $z \in Q_E^{A_1}$, $\gamma \in \Gamma$, and $e \in \Sigma$, we have that:

1) $h_{SE}^{A_1}(y,\gamma) = z \Rightarrow h_{SE}^{A_2}(y,\gamma) = z$ and
2) $h_{ES}^{A_1}(z,e) = y \Rightarrow h_{ES}^{A_2}(z,e) = y$.

B. Pruning Process

Example **II.1** shows how uncontrollable and uneditable events might potentially reveal the attacker's presence to the supervisor, by taking it to its "dead" state. Our ultimate objective is to construct a *stealthy* attack; thus we will prune the IDA to eliminate paths that reveal the presence of the attacker to the supervisor. The pruned IDA structure will only contain stealthy attacks.

The pruning process can be formulated similarly as a Basic Supervisory Control Problem (BSCP) as defined in [20]. However, we need to include an additional condition to ensure the non-existence of a race condition between the attacker and the system. Specifically, we do not allow the situation where the only possible move from the attacker is to insert an event; i.e., it cannot be the case that letting the event through and erasing it are both absent as possible moves of the attacker. Adding this extra condition to the standard algorithm for solving BSCP results in Algorithm 1.

Algorithm 1 Modified BSCP

Require: $A = (Q_E \cup Q_S, E, f_{se} \oplus f_{es}, a_0)$, where $E \subseteq (\Sigma_o \cup \Sigma_a^e \cup \Gamma)$ and $A_{trim} = (A^t, E, f^t, a_0^t)$, where $A^t \subseteq Q_E \cup Q_S$

Step 1 Compute $H_0 = (A_0, E, g_0, (a_0, a_0^t)) = A \times A_{trim}$, where $\times$ is the product of automata operation and $A_0 \subseteq (Q_S \cup Q_E) \times (Q_S \cup Q_E)$, and set $i = 0$

Step 2 Calculate

Step 2.1 $A_i' = \{(a,a^t) \in Y_i | \Gamma_A(\{a\}) \cap E_{uc} \subseteq \Gamma_{H_i}(\{(a,a^t)\})\}$

Step 2.2 $A_i'' = \{(a,a^t) \in Y_i' |\ e \in \Sigma_a \wedge e \in \Gamma_A(\{a\}) \rightarrow (e \in \Gamma_{H_i}(\{(a,a^t)\}) \vee e_d \in \Gamma_{H_i}(\{(a,a^t)\})\}$
$g_i' = g_i | A_i''$, transition function update

Step 2.3 $H_{i+1} = Trim(A_i'', E, g_i', (a_0, a_0^t))$

Step 3 If $H_{i+1} = H_i$, Stop; otherwise $i \leftarrow i+1$

Remark: The only difference between the original BSCP algorithm [20] and its modified version in Algorithm 1 is the addition of step 2.2 in the iteration process. Namely, any feasible system event must either be let through or erased, in addition to possible insertions by the attacker. That is, states where both the "let through" transition and the "erasure" transition are absent for a feasible system event will be deleted, as such a situation means that the attacker is forced to race to do insertions before the system executes that event.

Thus, to compute stealthy IDA we define as system the IDA constructed according to Definition **IV.2**. Moreover, any event in $e \in \Sigma_a \cup \Sigma_a^e$ is treated as *controllable* while events $e \in \Sigma_o \backslash \Sigma_a$ and $\gamma \in \Gamma$ are treated as *uncontrollable*. The specification language is obtained by deleting the states where the supervisor reaches the dead state, i.e., by deleting in IDA all states of the form $y = (S, \text{dead})$ for any $S \subseteq X$. We are able to consider all events in Σ_a as controllable since from the attacker's point of view, it could choose to prevent updates of the supervisor's state by erasing such events. In other words, the attacker is not "disabling any feasible uncontrollable event" in G, but rather it is suppressing the information given to the supervisor.

Let us now formalize the pruning process for obtaining all stealthy insertion-deletion attacks as follows.

Definition IV.3. Given the IDA A constructed according to Definition **IV.2**, define the system automaton $A_G = A$ with $\Sigma_c^A = \Sigma_a \cup \Sigma_a^e$ as the set of controllable events and $\Sigma_{uc}^A = (\Sigma_o \backslash \Sigma_a) \cup \{\gamma |\ \gamma \in \Gamma\}$ as the set of uncontrollable events. The specification automaton is defined by A_{trim}, which is obtained by trimming from A_G all its states of the form (S, dead), for any $S \subseteq X$. The *Stealthy IDA structure*, denoted by A_S, is defined to be the automaton obtained after running Algorithm 1 on A_{trim} w.r.t. A_G and Σ_a.

Lemma IV.1. If f_A satisfies conditions (1) and (2) of Problem **III.1**, then f_A can be synthesized from A_S.

Proof. We do a sketch of the proof. Given an attacker function f_A that satisfies conditions 1 and 2 from Problem **III.1**, it must be included in A_S, because A_S is constructed based on the IDA A from definition **IV.2**. The construction of A exhausts all possible feasible transitions in each E-state, therefore it is the maximal structure. When we build A_S, the only states deleted are those that lead the supervisor to its dead state. The deletion process does not remove any path that satisfies conditions 1 and 2. Therefore, all paths of f_A must exist in A_S, which means that f_A can be synthesized from A_S. □

Lemma IV.2. If an attacker function f_A is synthesized from A_S, then conditions (1) and (2) from Problem **III.1** are satisfied.

Proof. We do a sketch of the proof. Given an attacker function f_A synthesized from the A_S, we want to prove the lemma by contradiction. Thus, assume that f_A violates condition 1 or condition 2. First, let us assume only condition 2 is violated. If condition 2 is violated, there must exist a string $s \in \mathscr{L}(G)$ s.t. $S_A(P(s)) =$ dead. But, by construction, A_S does not contain any S-state $y = (y_1,\text{dead})$. Condition 1 on the other hand is never violated given that the construction of A_S specifies at each E-state all feasible transitions of the system S_A/G. Therefore, f_A must satisfy conditions 1 and 2 from Problem **III.1**. □

Lemma IV.3. The All Stealthy IDA A_S has all possible stealthy insertion-deletion attacks with respect to Σ_a, R and G.

Proof. The proof follows from Lemmas **IV.1** and **IV.2**. □

Example IV.3. Let us continue Example **IV.2** and construct the IDA A_S for the IDA A shown in Fig. 4(a). We obtain the specification to be used for Alg. 1 operation by deleting the red states in Fig. 4(a). The result of the Algorithm 1 is shown in Fig. 4(a), where states marked with a red cross were deleted. In essence, the result says that in order to remain stealthy, the attacker should not insert event b when it knows that the system is in state 0 and the supervisor is in state A. Note that, state $(\{2\},A,\{b\})$ is not deleted by Alg. 1, thus we can find a successful attack strategy. One attack strategy is to insert event b when the system is in state 1 and the supervisor is in state B, otherwise it lets the events reach the supervisor intact.

C. Analysis

The stealthy IDA structure A_S embeds all possible insertion-deletion actions that an attacker may perform while remaining unnoticed by the supervisor. Nevertheless, it is possible that none of these actions performed by the attacker will result in the system reaching a critical state. In this subsection, we provide some remarks about the stealthy IDA structure, along with the main theorem that addresses our synthesis Problem **III.1**.

First, let us provide a theorem based on the A_S structure.

Theorem IV.1. If the stealthy IDA structure A_S satisfies $A_S \sqsubseteq A_\emptyset$, where $A_\emptyset$ is the IDA structure given $\Sigma_a = \emptyset$, then there does not exist any f_A w.r.t. Σ_a that solves Problem **III.1**.

Proof. The result follows from Lemma **IV.3** and that the attacker cannot perform any action. □

Theorem **IV.1** only provides a necessary condition for the existence of a successful stealthy insertion-deletion attack. Thus, even if the A_S is a strictly bigger system than $A_\emptyset$, there is no guarantee of existence of a successful attack. Before we introduce the main theorem, let us add remarks about state properties related with A_S.

Orthogonal to the strong versus weak distinction in Problem **III.1**, attacks can also be classified along a different axis: *critical* and *deadlock* attacks. When the attacker successfully induces the system into reaching a critical unsafe state, we call this a *critical attack*. Alternatively, the attacker may attempt to induce the system into entering a deadlock; this is called a *deadlock attack*. Formally:

Definition IV.4. *Strong critical attacks* are defined to be the set of actions performed by the attacker such that A_S reaches E-state $z \in Q_E$ s.t. $\forall x \in I_1(z)$, $x \in X_{crit}$. Similarly, *weak critical attacks* are defined to be the set of actions performed by the attacker such that A_S reaches an E-state $z \in Q_E$ s.t. $\exists x \in I_1(z)$, $x \in X_{crit}$.

Definition IV.5. *Strong deadlock attacks* are defined to be the set of actions performed by the attacker such that A_S reaches E-state $z \in Q_E$ s.t. $\forall e \in \Gamma(z)$, $Next_e(I_1(z)) = \emptyset$. Similarly, *weak deadlock attacks* are defined to be the set of actions performed by the attacker such that A_S reaches an E-state $z \in Q_E$ s.t. $\exists x \in I_1(z)$ and $\forall e \in \Gamma(z)$, $Next_e(\{x\}) = \emptyset$.

Lastly, we say an attack is unsuccessful if it results in neither a deadlock nor a critical state.

Theorem IV.2. Given the system G, the supervisor R, and Σ_a, there exists f_A that strongly satisfies Problem **III.1** if and only if there exists a strong critical attack in the A_S structure. On the other hand, it weakly satisfies Problem **III.1** if and only if there exists a weak critical attack in the A_S structure and there does not exist a strong critical attack.

Proof. The proof follows from Lemma **IV.3**. □

Remark In Definition **IV.5**, deadlock attacks are introduced. An attacker could also identify deadlocks as harmful, with that in mind Problem **III.1** could be extended to capture deadlocks as goals for the attacker. Clearly, Theorem **IV.2** could be extended to that new problem.

D. Complexity Analysis

Given G with $|X|$ states, the current state estimator has at most $X_S = 2^{|X|}$. To build the IDA, as was mentioned before, it is necessary to maintain information of the state estimator of G as well as the current state of supervisor $\tilde{R}$. Therefore the IDA structure has the size $|X_S| \times |\tilde{Q}|$ in the worst case. In all, the space complexity of the IDA is $O(|X_S| \times |\tilde{Q}|)$.

V. Case Study: Water Treatment Plant

The Secure Water Treatment (SWaT) system[1] is a testbed located at the Singapore University of Technology and Design (SUTD). The system is a fully functional, scaled-down version of an industrial plant, and it performs all of the critical operations that are involved in a standard water treatment process. The system is associated with a number of safety requirements that it must satisfy, e.g., it must always maintain the level of water in its tank below a certain threshold. In this case study, we constructed a model of SWaT, and used a verification tool based on a first-order constraint solver to automatically generate stealthy attacks on the model using the framework presented in this work. The output of our synthesis approach describes realistic attacks that were successfully demonstrated on SWaT in a prior work [18], which does not provide a general formulation of stealthy attacks as we have done in this paper. Using our model, we generated several potential attacks that may cause the water tank to overflow. Details of the SWaT case study were omitted due to limited space, but can be found in [22].

VI. Conclusion

We have considered the problem of discovering stealthy deception attacks that cause physical damage in CPS. We define the attacker as an edit function that reacts to the plant's outputs and transforms them in a way that guarantees stealthiness. The IDA structure was introduced to capture the game between the environment (i.e., system and attacker) and the given supervisor. The IDA embeds all valid actions of the attacker. Based on the IDA, an algorithm was provided to synthesize a stealthy IDA that embeds all stealthy actions of an attacker. Given the stealthy IDA, we showed how to verify if there exists an edit function that leads the system to unsafe critical states without detection of the attack. In contrast to related work, we do not make any assumptions about the partitions of observable and controllable events of the system. We also described a case study validating our approach on a realistic example.

In the future, we plan to investigate how to modify a supervisor that is susceptible to stealthy deception attacks. We also plan to tackle the problem of designing directly supervisors that enforce safety and liveness specifications and at the same time are robust to deception attacks.

VII. Acknowledgements

The authors gratefully acknowledge the contributions of Xiang Yin and Christoforos Keroglou in reviewing the paper and providing insightful comments. We would also like to thank Sridhar Adepu and Aditya P. Mathur at the Singapore University of Technology and Design for their help with the case study on the water treatment system. Finally, we thank an anonymous reviewer for pointing out the need to add step 2.2 in Algorithm 1.

[1] https://itrust.sutd.edu.sg/research/testbeds/secure-water-treatment-swat

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*, June 2008, pp. 495–500.

[2] 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*, ser. HiCoNS '12. New York, NY, USA: ACM, 2012, pp. 55–64.

[3] D. Thorsley and D. Teneketzis, "Intrusion detection in controlled discrete event systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec 2006, pp. 6047–6054.

[4] A. Paoli, M. Sartini, and S. Lafortune, "Active fault tolerant control of discrete event systems using online diagnostics," *Automatica*, vol. 47, no. 4, pp. 639–649, Apr. 2011.

[5] L. K. Carvalho, Y. C. Wu, R. Kwong, and S. Lafortune, "Detection and prevention of actuator enablement attacks in supervisory control systems," in *13th International Workshop on Discrete Event Systems (WODES)*, May 2016, pp. 298–305.

[6] M. Wakaiki, P. Tabuada, and J. P. Hespanha, "Supervisory control of discrete-event systems under attacks," *CoRR*, vol. abs/1701.00881, 2017. [Online]. Available: http://arxiv.org/abs/1701.00881

[7] R. Su, "Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations," *CoRR*, vol. abs/1608.04103, 2016. [Online]. Available: http://arxiv.org/abs/1608.04103

[8] K. Rohloff, "Bounded sensor failure tolerant supervisory control," *IFAC Proceedings Volumes*, vol. 45, no. 29, pp. 272 – 277, 2012.

[9] F. Lin, "Control of networked discrete event systems: Dealing with communication delays and losses," *SIAM Journal on Control and Optimization*, vol. 52, no. 2, pp. 1276–1298, 2014.

[10] M. V. S. Alves, J. C. Basilio, A. E. C. da Cunha, L. K. Carvalho, and M. V. Moreira, "Robust supervisory control against intermittent loss of observations," *IFAC Proceedings Volumes*, vol. 47, no. 2, pp. 294 – 299, 2014.

[11] X. Yin, "Supervisor synthesis for mealy automata with output functions: A model transformation approach," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, 2016.

[12] A. Saboori and C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," in *46th IEEE Conference on Decision and Control*, Dec 2007, pp. 5056–5061.

[13] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, Mar. 2011.

[14] F. Cassez, J. Dubreil, and H. Marchand, "Synthesis of opaque systems with static and dynamic masks," *Formal Methods in System Design*, vol. 40, no. 1, pp. 88–115, 2012.

[15] Y.-C. Wu, V. Raman, B. C. Rawlings, S. Lafortune, and S. A. Seshia, "Synthesis of obfuscation policies to ensure privacy and utility," *Journal of Automated Reasoning*, Jul 2017. [Online]. Available: https://doi.org/10.1007/s10817-017-9420-x

[16] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2140–2154, Aug 2016.

[17] ——, "Synthesis of maximally-permissive supervisors for the range control problem," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, Dec. 2016.

[18] E. Kang, S. Adepu, D. Jackson, and A. P. Mathur, "Model-based security analysis of a water treatment system," in *Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems, SEsCPS@ICSE 2016, Austin, Texas, USA, May 14-22, 2016*, 2016, pp. 22–28.

[19] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, Jan. 1987.

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

[21] A. Paoli and S. Lafortune, "Safe diagnosability for fault-tolerant supervision of discrete-event systems," *Automatica*, vol. 41, no. 8, pp. 1335–1347, Aug. 2005.

[22] R. M. Goes, R. Kwong, E. Kang, and S. Lafortune, "Stealthy deception attacks for cyber-physical systems," Department of Electrical Engineering and Computer Science, Ann Arbor, Michigan, Tech. Rep., August 2017.