Synthesis of Actuator Attackers for Free

Liyong Lin · Yuting Zhu · Rong Su

Received: date / Accepted: date

Abstract In this work, we shall formulate and address a problem of actuator attacker synthesis for cyber-physical systems that are modeled by discrete-event systems. We assume the actuator attacker partially observes the execution of the closed-loop system and it can modify each control command generated by the supervisor on a specified attackable subset of the controllable events. We provide straightforward reductions from the actuator attacker synthesis problems to the Ramadge-Wonham supervisor synthesis problems. It then follows that it is possible to use the many techniques and tools already developed for solving the supervisor synthesis problem to solve the actuator attacker synthesis problem for free. In particular, we show that, if the attacker cannot attack unobservable events to the supervisor, the reductions can be carried out in polynomial time.

Keywords cyber-physical systems · discrete-event systems · supervisory control · actuator attack · partial observation

1 Introduction

Recently, security of cyber-physical systems has drawn much research interest within the discrete-event systems and formal methods community [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17]. For a recent survey on the discrete-event systems based approach for the security of cyber-physical systems, the reader is referred to [18]. In this paper, we shall focus on discrete-event systems as our model of cyber-physical systems and consider the problem of attacker synthesis, as a major step towards solving the resilient supervisor synthesis problem [14]. In particular, we consider the synthesis of a successful actuator attacker [12], [14]. We assume the goal of the actuator attacker is to remain covert in the process of attacking the closed-loop systems until it causes damages, much in parallel to the framework of [3]. An actuator

Liyong Lin, Yuting Zhu, Rong Su
Electrical and Electronic Engineering, Nanyang Technological University, Singapore
E-mail: liyong.lin@ntu.edu.sg, yuting002@e.ntu.edu.sg, rsu@ntu.edu.sg
attacker is said to be successful if it achieves its goal. Correspondingly, a supervisor is said to be resilient if a successful actuator attacker does not exist.

Some of the existing works consider the problem of deception attack [3], [4], [7], where the attacker would accomplish the attack goal by altering the sensor readings. There are also works that deal with actuator attacks as well as sensor attacks [2], [8], [9], [15]. We provide some comparisons between the existing works and this work in the following.

1. The works of [2] and [8] do not require the attacker to possess any knowledge of the system’s models; consequently, they consider the worst case attack scenario (attack the system whenever possible), and thus the problem of attacker synthesis becomes irrelevant in their problem setup. In contrast, we consider attackers that know the models of the system, including the plant’s and the supervisor’s models. This allows the attacker to make informed attack decisions and makes it possible to model different goals and constraints of the attackers. In particular, this allows us to model the synthesis of risky attackers and covert attackers [12], [14]. Moreover, it is worth noting that imposing this assumption on the attacker’s knowledge does not bring any harm. A supervisor that can guard against damages caused by knowledgeable attackers can certainly also guard against “ignorant” attackers, if we do not impose covertness assumption on knowledgeable attackers.

2. The only means of system protection in the setup of [2] and [8] is by making use of an intrusion detection module for detecting the presence of an attacker, along with a fixed supervisor that may not have been properly designed against actuator attackers. Their setup does not consider the possibility of modifying the supervisor to render attack impossible or harder. In comparison, the resilient supervisor synthesis based approach in [3], [14] may allow the synthesis of a supervisor that can prevent the damages caused by attackers, by making attack impossible or by making attack futile, even in the absence of a monitor, or by discouraging attack attempt if covertness assumption is imposed, in the presence of a monitor [12]. In particular, [2] and [8] only address the problem of verification, instead of synthesis, for security.

3. The work of [15] is more closely related to our work and is based on the resilient supervisor synthesis framework. It attempts to present a generalization of [3] that uses non-deterministic finite state transducers as the unifying models and address the security against both actuator attackers and sensor attackers. However, it only deals with the verification of resilient supervisor upon newly defined controllability condition. The problem of synthesis of resilient supervisor remains unsolved if the controllability condition is not satisfied. We note that solving the problem of attacker synthesis in this work naturally solves the resilient supervisor verification problem and can be used for solving the resilient supervisor synthesis problem as well, as explained in [14].

4. In [12], [13] and [14], we have considered partially-observing attackers that also eavesdrop the control commands sent by the supervisor. Given the same observa-
tion capability over the plant events, the control command eavesdropping attacker is more powerful than the attacker considered in this work. In [17], a notion of a powerful attacker is defined for passive attackers. In particular, a passive attacker is defined to be powerful in [17] if it can directly query some information about the current state of the plant. A control command eavesdropping attacker in [12] can be considered a powerful attacker in the sense of [17], but in a more indirect but arguably a more realistic way. In particular, the supervisor’s control command does encode some knowledge of the partially-observing supervisor on the execution of the plant. We would expect that the technique used for synthesizing non-eavesdropping attackers can be extended to synthesizing control command eavesdropping attackers.

In this work, we shall focus on the actuator attacker synthesis problem. The paper is organized as follows: Section 2 is devoted to the preliminaries. Then, in Section 3, we shall provide a detailed explanation about the system setup, including a formulation of the supervisor, the actuator attacker, the damage automaton, the monitoring mechanism, the attacked closed-loop system and the attack goal. In Section 4, we then provide straightforward reductions from the actuator attacker synthesis problems to the supervisor synthesis problems. Finally, discussions and conclusions are presented in Section 5.

2 Preliminaries

We assume the reader is familiar with the basics of supervisory control theory [19], [20] and automata theory [21]. In this section, we shall recall the basic notions and terminology that are necessary to understand this work.

For any two sets A and B, we use A × B to denote their Cartesian product and use A − B to denote their set difference. |A| is used to denote the cardinality of set A.

A (partial) finite state automaton $G$ over alphabet $\Sigma$ is a 5-tuple $(Q, \Sigma, \delta, q_0, Q_m)$, where $Q$ is the finite set of states, $\delta : Q \times \Sigma \rightarrow Q$ the partial transition function, $q_0 \in Q$ the initial state and $Q_m \subseteq Q$ the set of marked states. Let $L(G)$ and $L_m(G)$ denote the closed-behavior and the marked-behavior of $G$, respectively [19]. When $Q_m = Q$, we also write $G = (Q, \Sigma, \delta, q_0)$ for simplicity. For any two finite state automata $G_1 = (Q_1, \Sigma_1, \delta_1, q_{1,0}, Q_{1,m}), G_2 = (Q_2, \Sigma_2, \delta_2, q_{2,0}, Q_{2,m})$, we write $G := G_1 || G_2$ to denote their synchronous product. Then, we have that $G = (Q := Q_1 \times Q_2, \Sigma := \Sigma_1 \cup \Sigma_2, \delta := \delta_1 \times \delta_2, q_0 := (q_{1,0}, q_{2,0}), Q_m := Q_{1,m} \times Q_{2,m})$, where the (partial) transition function $\delta$ is defined as follows: for any $q = (q_1, q_2) \in Q$ and any $\sigma \in \Sigma$,

$$\delta(q, \sigma) := \begin{cases} (\delta_1(q_1, \sigma), q_2), & \text{if } \sigma \in \Sigma_1 \setminus \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)), & \text{if } \sigma \in \Sigma_2 \setminus \Sigma_1 \\ (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \end{cases}$$

2 We write $\delta(q, \sigma)$ to mean $\delta(q, \sigma)$ is defined and write $\neg \delta(q, \sigma)$ to mean $\delta(q, \sigma)$ is undefined. As usual, $\delta$ is naturally extended to the partial transition function $\delta : Q \times \Sigma^* \rightarrow Q$ such that, for any $q \in Q$, any $s \in \Sigma^*$ and any $\sigma \in \Sigma$, $\delta(q, \sigma)$ is defined and $\delta(q, \sigma) = \delta(q, \sigma s) = \delta(q, \sigma)$. We define $\delta(q', \sigma) = \{\delta(q, \sigma) \mid q \in Q'\}$ for any $Q' \subseteq Q$.

3 For example, if $\sigma \in \Sigma_1 \setminus \Sigma_2$ and $\delta_1(q_1, \sigma)$ is undefined, we treat $\delta(q, \sigma)$ as undefined.
A control constraint over $\Sigma$ is a tuple $(\Sigma_c, \Sigma_o)$ of sub-alphabets of $\Sigma$, where $\Sigma_o \subseteq \Sigma$ denotes the subset of observable events (for the supervisor) and $\Sigma_c \subseteq \Sigma$ denotes the subset of controllable events (for the supervisor). Let $\Sigma_{no} = \Sigma - \Sigma_o \subseteq \Sigma$ denote the subset of unobservable events (for the supervisor) and let $\Sigma_{nc} = \Sigma - \Sigma_c \subseteq \Sigma$ denote the subset of uncontrollable events (for the supervisor). For each sub-alphabet $\Sigma' \subseteq \Sigma$, the natural projection $P_{\Sigma'} : \Sigma^* \to \Sigma'^*$ is defined, which is extended to a map between languages as usual [19]. Let $G = (Q, \Sigma, \delta, q_0)$. We shall abuse the notation and define $P_{\Sigma'}(G)$ to be the finite state automaton $(2^Q, \Sigma, \Delta, U_{RG, \Sigma-L'}(q_0))$, where the unobservable reach $U_{RG, \Sigma-L'}(q_0) := \{ q \in Q \mid \exists s \in (\Sigma - \Sigma')^*, q = \delta(q_0, s) \}$ of $q_0$ with respect to the sub-alphabet $\Sigma - \Sigma' \subseteq \Sigma$ is the initial state, and the partial transition function $\Delta : 2^Q \times \Sigma \longrightarrow 2^Q$ is defined as follows.

1. For any $\emptyset \neq Q' \subseteq Q$ and any $\sigma \in \Sigma'$, $\Delta(Q', \sigma) = U_{RG, \Sigma-L'}(\delta(Q', \sigma))$, where $U_{RG, \Sigma-L'}(Q') := \bigcup_{q \in Q'} U_{RG, \Sigma-L'}(q)$ for any $Q' \subseteq Q$.
2. for any $\emptyset \neq Q' \subseteq Q$ and any $\sigma \in \Sigma - \Sigma'$, $\Delta(Q', \sigma) = Q'$.

In particular, we shall note that $P_{\Sigma'}(G)$ is over the alphabet $\Sigma$ and there is no transition defined at the state $\emptyset \in 2^Q$. A finite state automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$ is said to be non-blocking if every reachable state in $G$ can reach a marked state in $Q_m$.

3 System Setup

In this section, we shall introduce the system setup that will be followed in this work. Intuitive explanations about the basic ideas and assumptions are provided in Section 3.1. After that, a formal setup is provided in Section 3.2.

3.1 Basic Ideas

The plant $G$, which is a finite state automaton over $\Sigma$, is under the control of a partially observing supervisor $S$ over $(\Sigma_c, \Sigma_o)$, which is also given by a finite state automaton over $\Sigma$. In addition, we assume the existence of an actuator attacker $A$ that partially observes the execution of the closed-loop system and is able to modify the control command $\gamma$ (generated by the supervisor) on a specified attackable subset $\Sigma_{c,A} \subseteq \Sigma_c$ of the controllable events. The plant $G$ then follows the modified control command $\gamma'$ instead of $\gamma$. We assume the supervisor $S$ is augmented with a monitoring mechanism that monitors the execution of the closed-loop system (under attack). If $\sigma \in \Sigma_{c,A}$ is enabled by the supervisor in $\gamma$, then the attacker is not discovered by the supervisor when it disables $\sigma$ in $\gamma'$. The supervisor can conclude the existence of an attacker the first moment when it observes an inconsistency between what it has observed and what shall be observed without an attacker. We assume that the attacker has a complete knowledge about the models of the plant $G$ and the supervisor $S$, as well as the control constraint $(\Sigma_c, \Sigma_o)$. Intuitively, the goal of the attacker is to remain
covert in the process of attacking the closed-loop systems until it causes damages, by generating certain damaging strings. The supervisor has a mechanism for halting the execution of the closed-loop system after discovering an actuator attack. We shall assume no communication delay is involved for the sending and receiving of observable events and control commands.

3.2 Formal Setup

In this section, we explain the formal setup that is used in this work. We first need to introduce and present a formalization of the system components.

**Supervisor:** In the absence of attacker, a supervisor over control constraint \((\Sigma, \Sigma_i)\) is modeled by a finite state automaton \(\mathcal{S} = (X, \Sigma, \zeta, x_0)\) that satisfies the controllability and observability constraints \([22]\):

- (controllability) for any state \(x \in X\) and any uncontrollable event \(\sigma \in \Sigma_{\text{uc}}, \zeta(x, \sigma)\), implies \(\zeta(x, \sigma) = x\),
- (observability) for any state \(x \in X\) and any unobservable event \(\sigma \in \Sigma_{\text{om}}, \zeta(x, \sigma)\),

**Plant:** The plant is modeled by a finite state automaton \(\mathcal{G} = (Q, \Sigma, \delta, q_0)\) as usual. Whenever the plant fires an observable transition \(\delta(q, \sigma) = q'\), it sends the observable event \(\sigma\) to the supervisor.

**Actuator Attacker:** We assume the attacker knows the model of the plant \(\mathcal{G}\) and the model of the supervisor \(\mathcal{S}\). We impose some restrictions on the capability of the attacker. Let \(\Sigma_{\text{v}, A} \subseteq \Sigma\) denote the subset of (plant) events that can be observed by the attacker. Let \(\Sigma_{\text{c}, A} \subseteq \Sigma\) denote the set of attackable events. The attacker can modify the control command \(\gamma\) generated by the supervisor on the subset \(\Sigma_{\text{c}, A}\).

We henceforth name \((\Sigma_{\text{v}, A}, \Sigma_{\text{c}, A})\) as an attack constraint. An actuator attacker over attack constraint \((\Sigma_{\text{v}, A}, \Sigma_{\text{c}, A})\) is modeled by a finite state automaton \(\mathcal{A} = (Y, \Sigma, \beta, y_0)\) that satisfies the following constraints.

- (A-controllability) for any state \(y \in Y\) and any unattackable event \(\sigma \in \Sigma - \Sigma_{\text{c}, A}, \beta(y, \sigma)\),
- (A-observability) for any state \(y \in Y\) and any unobservable event \(\sigma \in \Sigma - \Sigma_{\text{v}, A}\) to the attacker, \(\beta(y, \sigma)\) implies \(\beta(y, \sigma) = y\).

Compared with \([12]\) and \([3]\), here we have adopted a unifying model for the modeling of the plant, the supervisor and the attacker. In particular, it is straightforward to see that an attacker \(A\) over attack constraint \((\Sigma_{\text{v}, A}, \Sigma_{\text{c}, A})\) can also be viewed as a supervisor over control constraint \((\Sigma_{\text{c}, A}, \Sigma_{\text{v}, A})\).

**Damage Automaton:** To specify in a general manner what strings can cause damages, we shall adopt a complete finite state automaton \(\mathcal{H} = (W, \Sigma, \chi, w_0, W_m)\) \([12]\). In particular, each string \(s \in L_m(H)\) is a damage-inflicting string that the attacker would like the attacked closed-loop system to generate. In the special case of state avoidance property, we can get rid of \(H\) and introduce the set \(Q_{\text{bad}} \subseteq Q\) of bad states to avoid in the plant \(\mathcal{G}\). Without loss of generality, we shall make the assumption that each \(w \in W_m\) is a sink state, i.e., \(\forall \sigma \in \Sigma, w \in W_m, \chi(w, \sigma) = w\). Intuitively, this means that the damage is never recoverable. Thus, we can assume \(|W_m| = 1\), i.e., there is only one sink state, without loss of generality. Let \(\mathcal{H} = (W, \Sigma, \chi, w_0, \{w_m\})\) in the rest.
Monitoring Mechanism: In the presence of an attacker, we assume the supervisor is augmented with a monitoring mechanism. We assume, without loss of generality, that the execution of the attacked closed-loop system is immediately halted once the supervisor discovers the presence of an attacker. The supervisor online records its observation \( w \in \Sigma^* \) of the execution of the (attacked) closed-loop system. It concludes the existence of an attacker and then halts the execution of the (attacked) closed-loop system the first time when it observes some string \( w \notin P_{Gc}(L(S||G)) \).

Attacked Closed-loop System: The attacked closed-loop system consists of the three main components, \( G, S \) and \( A \). However, the synchronous product of \( G, S \) and \( A \) is not the attacked closed-loop system for the following two reasons.

1. The effect of attack on the supervisor is not reflected in \( S \)
2. The monitoring mechanism has not been encoded in \( S \)

The first problem can be solved by computing the (fully) attacked supervisor \( S^A \). Let \( S = (X, \Sigma, \zeta, x_0) \). We let \( S^A = (X \cup \{x_{halt}\}, \Sigma, \zeta^A, x_0) \) denote the attacked supervisor, where \( x_{halt} \notin X \) denotes a distinguished halting state and the partial transition function \( \zeta^A \) is obtained from \( \zeta \) as follows.

a) for any \( x, x' \in X \) and any \( \sigma \in \Sigma \), if \( \zeta(x, \sigma) = x' \), then \( \zeta^A(x, \sigma) = x' \)
b) for any \( x \in X \) and any \( \sigma \in \Sigma_{\text{u}} \cap \Sigma_{\text{c}A} \), if \( \neg \zeta(x, \sigma) \), then \( \zeta^A(x, \sigma) = x \)
c) for any \( x \in X \) and any \( \sigma \in \Sigma_{\text{u}} \cap \Sigma_{\text{c}A} \), if \( \neg \zeta(x, \sigma) \), then \( \zeta^A(x, \sigma) = x_{halt} \)

Intuitively, to obtain \( S^A \), for each state \( x \in X \) and each event \( \sigma \in \Sigma_{\text{u}} \cap \Sigma_{\text{c}A} \), if \( \sigma \) is not defined at state \( x \), we then add a self-loop at state \( x \) labeled by \( \sigma \); for each state \( x \in X \) and each event \( \sigma \in \Sigma_{\text{u}} \cap \Sigma_{\text{c}A} \), if \( \sigma \) is not defined at state \( x \), then we add a transition from \( x \) to the halting state \( x_{halt} \) labeled by \( \sigma \). In particular, we note that there is no transition defined at state \( x_{halt} \) and every state in \( S^A \) is a marked state.

With \( S^A \), we effectively assume an ignorant and risky attacker, in the sense of [2] and [3], that enables each attackable event at each state. Then, the rest of the attack decisions for the actuator attacker is to disable attackable events properly, based on its own partial observation and its knowledge on the system’s models, to ensure covertness and damage-infliction. This then brings the actuator attacker synthesis problem closer to the supervisor synthesis problem.

The second problem has been partially solved in the solution to the first problem. In particular, if \( S^A \) enters \( x_{halt} \), then we know that attack must have happened. This is insufficient, however. Recall that the supervisor concludes the existence of an attacker and then halts the execution of the (attacked) closed-loop system the first time when it observes some string \( w \notin P_{Gc}(L(S||G)) \). We now denote \( S||G = (X \times Q, \Sigma, \lambda, (x_0, q_0)) \), where \( \lambda = \zeta || \delta \). Then, we have

\[
P_{Gc}(S||G) = (2^{X \times Q}, \Sigma, \Lambda, UR_{S||G} \Sigma \Sigma_e(x_0, q_0)),
\]

where \( UR_{S||G} \Sigma \Sigma_e(x_0, q_0) \) denotes the unobservable reach of \( (x_0, q_0) \) with respect to the sub-alphabet \( \Sigma \) in \( S||G \) and the partial transition function \( \Lambda : 2^{X \times Q} \times \Sigma \rightarrow 2^{X \times Q} \) is defined such that,

\(^5\) In [5], when the supervisor detects the presence of an attacker, all controllable events will be disabled, while uncontrollable events can still occur (i.e., immediate halt by reset is impossible). This is not difficult to accommodate, by using a device \( H \) that defines the set of damage-inflicting strings [12]. In particular, if uncontrollable events can still occur after the detection, then we only need to use \( H' = H \cup H'_{\Sigma_u} \), where \( H'_{\Sigma_u} := \{ x \in \Sigma^* | \exists x' \in \Sigma_{\text{c}u}, x' \notin H \} \), as the renewed set of damage-inflicting strings.
1. for any \( \varnothing \neq D \subseteq X \times Q \) and any \( \sigma \in \Sigma_o \), \( \Lambda(D, \sigma) = U R_{S\parallel G, \Sigma - \Sigma_o} (\hat{\lambda}(D, \sigma)) \);
2. for any \( \varnothing \neq D \subseteq X \times Q \) and any \( \sigma \in \Sigma - \Sigma_o \), \( \Lambda(D, \sigma) = D \).

Intuitively, we can use \( P_{\Sigma_o}(S\parallel G) \) to track the belief \( \varnothing \neq D \subseteq X \times Q \) of the supervisor on the current states of the unattacked closed-loop system \( S\parallel G \), for monitoring purpose. When the current belief of the supervisor is \( \varnothing \neq D \subseteq X \times Q \) and \( \sigma \in \Sigma_o \) is executed with \( \hat{\lambda}(D, \sigma) = \varnothing \), the supervisor detects an inconsistency and determines that attack must have already happened and thus halts the execution of the attacked closed-loop system. In particular, we shall note that there is no transition defined at state \( \varnothing \in 2^{X \times Q} \) and every state in \( P_{\Sigma_o}(S\parallel G) \) is a marked state.

Given the plant \( G \), the supervisor \( S \) and the attacker \( A \), the attacked closed-loop system is the synchronous product \( G\parallel S\parallel P_{\Sigma_o}(S\parallel G)\parallel A \), which is a finite state automaton over \( \Sigma \). Since the set of states the supervisor believes itself to be residing in is a singleton (due to the observability constraint for a supervisor) and corresponds to the state that it actually resides in, whenever \( P_{\Sigma_o}(S\parallel G) \) is not in the state \( \varnothing \), the state space of \( G\parallel S\parallel P_{\Sigma_o}(S\parallel G)\parallel A \), when restricted to the reachable state set, is isomorphic to \( Q \times (X \times \{x_{\text{halt}}\}) \times 2^Q \times Y \). In the rest, we consider \( Q \times (X \times \{x_{\text{halt}}\}) \times 2^Q \times Y \) as the state space.

**Successful Actuator Attackers:** In this work, we shall focus on covert attackers\(^6\). Given any plant \( G \), any supervisor \( S \), any actuator attacker \( A \) and any damage automaton \( H = (W, \Sigma, \lambda, w_0, \{w_m\}) \), the definition of covertness is given in the following.

**Definition 1 (Covertness)** Given any plant \( G \), any supervisor \( S \), any actuator attacker \( A \) and any damage automaton \( H \), \( A \) is said to be covert on \( (G, S) \) with respect to \( H \) if each state in \( \{(q, x, D_Q, y, w) \in Q \times (X \cup \{x_{\text{halt}}\}) \times 2^Q \times Y \times W \mid (x = x_{\text{halt}} \lor D_Q = \varnothing) \land w \neq w_m \} \) is not reachable in \( G\parallel S\parallel P_{\Sigma_o}(S\parallel G)\parallel A \parallel H \).

We shall consider two “possibilistic” interpretation of damaging goals for covert actuator attackers. The first interpretation of interest is the damage-nonblocking goal, which intuitively states that it is always possible to reach damages in the attacked closed-loop system, wherever the current state is for the attacked closed-loop system. That is, for any string \( s \in L(G) \parallel A \) generated by the attacked closed-loop system, it is always possible to extend it in \( G\parallel S\parallel P_{\Sigma_o}(S\parallel G)\parallel A \) to some string in \( L_m(H) \). This is formally captured by the next definition.

**Definition 2 (Damage-nonblocking)** Given any plant \( G \), any supervisor \( S \), any damage automaton \( H \) and any covert actuator attacker \( A \) on \( (G, S) \) with respect to \( H \), \( A \) is said to be damage-nonblocking on \( (G, S) \) with respect to \( H \) if \( G\parallel S\parallel P_{\Sigma_o}(S\parallel G)\parallel A \parallel H \) is non-blocking.

The second interpretation of interest for us is the damage-reachable goal\(^{12},^{13},^{14}\), which intuitively states that it is possible to reach damages in the attacked closed-loop system from the initial state. It is worth mentioning that, due to the existence of events that are beyond the control of an attacker, that is, any event \( \sigma \notin \Sigma_A \), this is arguably the weakest interpretation for damage-infliction goal. We here note that the set of marked states for \( G\parallel S\parallel P_{\Sigma_o}(S\parallel G)\parallel A \parallel H \) is \( \{(q, x, D_Q, y, w) \in Q \times (X \cup \{x_{\text{halt}}\}) \times 2^Q \times Y \times W \mid w = w_m \} \).

\(^6\) It is possible to model covert attackers and risky attackers in a unifying manner, see\(^{14}\).
Definition 3 (Damage-reachable) Given any plant $G$, any supervisor $S$, any damage automaton $H$ and any covert actuator attacker $A$ on $(G, S)$ with respect to $H$, $A$ is said to be damage-reachable on $(G, S)$ with respect to $H$ if some marked state in $G||S^A||P_{\Sigma_c}(S||G)||A||H$ is reachable, that is, $L_m(G||S^A||P_{\Sigma_c}(S||G)||A||H) \neq \emptyset$.

It is clear that the damage-nonblocking goal is stronger than the damage-reachable goal.

Proposition 1 Given any plant $G$ over $\Sigma$, any supervisor $S$ over control constraint $(\Sigma_r, \Sigma_o)$, any attack constraint $(\Sigma_o, \Sigma_i)$ and any damage automaton $H$ over $\Sigma$, if a covert actuator attacker $A$ over $(\Sigma_o, \Sigma_i)$ is damage-blocking on $(G, S)$ with respect to $H$, then $A$ is damage-reachable on $(G, S)$ with respect to $H$.

Proof. This follows straightforwardly from the definitions of the damage-nonblocking goal and the damage-reachable goal. In particular, $G||S^A||P_{\Sigma_c}(S||G)||A||H$ is non-blocking implies that some marked state in $G||S^A||P_{\Sigma_c}(S||G)||A||H$ is reachable. 

4 Actuator Attacker Synthesis

In this section, we provide straightforward reductions from the actuator attacker synthesis problems, with the two different attack goals, to the Ramadge-Wonham supervisor synthesis problems. It then follows that we can use the techniques and tools developed for solving the supervisor synthesis problem to solve the actuator attacker synthesis problem for free. We shall provide the reductions for the two different interpretations of damaging goals, separately. The details of the reductions are summarized in the next two theorems.

Theorem 1 Given any plant $G$ over $\Sigma$, any supervisor $S$ over control constraint $\mathcal{C} = (\Sigma_r, \Sigma_o)$, any attack constraint $\mathcal{A} = (\Sigma_o, \Sigma_i)$ and any damage automaton $H$ over $\Sigma$, there is a damage-nonblocking covert actuator attacker $A$ over $\mathcal{A}$ on $(G, S)$ with respect to $H$ iff there is a supervisor $S'$ over $\mathcal{C}' = (\Sigma_r, \Sigma_o)$ such that $S'||P$ avoids reaching the set $BAD$ of bad states in $P$ and $S'||P$ is non-blocking, where $P = G||S^A||P_{\Sigma_c}(S||G)||H$ and $BAD := \{(q, x, D_Q, w) \in Q \times (X \cup \{x_{halt}\}) \times 2^Q \times W \mid (x = x_{halt} \lor D_Q = \emptyset) \land w \neq w_m\}$.

Proof. This straightforwardly follows from the definition of a damage-nonblocking covert actuator attacker. In particular, $A$ is a damage-nonblocking covert actuator attacker over $\mathcal{A}$ on $(G, S)$ with respect to $H$ iff as a supervisor over $\mathcal{C}' = (\Sigma_r, \Sigma_o)$, $A||P$ avoids reaching the set $BAD$ of bad states in $P$ and $A||P$ is non-blocking, where $P = G||S^A||P_{\Sigma_c}(S||G)||H$ and $BAD := \{(q, x, D_Q, w) \in Q \times (X \cup \{x_{halt}\}) \times 2^Q \times W \mid (x = x_{halt} \lor D_Q = \emptyset) \land w \neq w_m\}$.

Theorem 2 Given any plant $G$ over $\Sigma$, any supervisor $S$ over control constraint $\mathcal{C} = (\Sigma_r, \Sigma_o)$, any attack constraint $\mathcal{A} = (\Sigma_o, \Sigma_i)$ and any damage automaton $H$ over $\Sigma$, there is a damage-reachable covert actuator attacker $A$ over $\mathcal{A}$ on $(G, S)$ with respect to $H$ iff there is a supervisor $S'$ over $\mathcal{C}' = (\Sigma_r, \Sigma_o)$ such that $S'||P$ avoids reaching the set $BAD$ of bad states in $P$ and $L_m(S'||P) \neq \emptyset$, where $P = G||S^A||P_{\Sigma_c}(S||G)||H$ and $BAD := \{(q, x, D_Q, w) \in Q \times (X \cup \{x_{halt}\}) \times 2^Q \times W \mid (x = x_{halt} \lor D_Q = \emptyset) \land w \neq w_m\}$. 

8 L. Lin, Y. Zhu, R. Su
Proof.  This straightforwardly follows from the definition of a damage-reachable covert actuator attacker. In particular, $A$ is a damage-reachable covert actuator attacker over $\mathcal{A}$ on $(G, S)$ with respect to $H$ iff as a supervisor over $\mathcal{A}' = (\Sigma_{cA}, \Sigma_A)$, $A \parallel P$ avoids reaching the set BAD of bad states in $P$ and $I_m(A \parallel P) \neq \emptyset$, where $P = G \parallel S^A \parallel P_{\Sigma}(S \parallel G) \parallel H$ and BAD := \{(q, x, D_q, w) \in Q \times (X \cup \{x_{halt}\}) \times 2^X \times W \mid q = x_{halt} \lor D_q = \emptyset \land w \neq w_m\}. \qed$

In general, the reductions provided in the above two theorems cannot be carried out in polynomial time. The main trouble is due to the need for tracking the state in $P_{\Sigma}(S \parallel G)$, whose state size is in general exponential in the state size of $G$ (modulo the state space of $S$). Under certain assumption on the set of attackable events, the above reductions are indeed polynomial time reductions, regardless of the models for $S$ and $G$. One example is given in the next result.

**Theorem 3** If $\Sigma_m \cap \Sigma_{cA} = \emptyset$, then the actuator attacker synthesis problem is polynomial time reducible to the supervisor synthesis problem, for both the damage-nonblocking goal and the damage-reachable goal.

**Proof.** Consider the attacked closed-loop system $G \parallel S^A \parallel P_{\Sigma}(S \parallel G) \parallel A$. We now restore the component $2^X \times Y$ for analysis; then a state of $G \parallel S^A \parallel P_{\Sigma}(S \parallel G) \parallel A$ is of the form $(q, x, D_q, y) \in Q \times (X \cup \{x_{halt}\}) \times 2^X \times Y$. Let $\text{dom}(D) := \{x \in X \mid \exists y \in Q, (x, q) \in D\}$. We have already known that $\text{dom}(D)$ is the singleton $\{x\}$, when restricted to the reachable state set of $G \parallel S^A \parallel P_{\Sigma}(S \parallel G) \parallel A$. Let $\text{Img}(D) := \{q \in D \mid \exists x \in X, (x, q) \in D\}$.

When the belief of the supervisor is $\emptyset \neq D \subseteq X \times Q$ and $\sigma \in \Sigma_{m}$ is executed with $\lambda_i(D, \sigma) = \emptyset$, the supervisor detects an inconsistency and determines that attack must have already happened. There are two possible reasons for $\lambda(D, \sigma) = \emptyset$. The first reason is when $\zeta(x, \sigma)$ is undefined for the unique state $x \in \text{dom}(D)$. This has already been taken care of by the component $S^A$. The second reason is when $\delta(\text{Img}(D), \sigma) = \emptyset$, which can be tracked with the component $P_{\Sigma}(S \parallel G)$. However, since $\sigma$ has been executed at the current plant state $q \in Q$, we conclude that $q \notin \text{Img}(D)$. That is, the current state the plant is in does not belong to the set of states the supervisor believes the plant to be residing in. This is only possible if some event $\sigma' \in \Sigma_m \cap \Sigma_{cA}$ has been enabled by the actuator attacker but disabled by the supervisor in the past execution. If $\Sigma_m \cap \Sigma_{cA} = \emptyset$, then the above possibility can be ruled out.

Thus, we can effectively get rid of the component $2^\ell$ (due to the use of $P_{\Sigma}(S \parallel G)$) in the construction of the attacked closed-loop systems, when $\Sigma_m \cap \Sigma_{cA} = \emptyset$. It follows that the reductions provided in Theorem 1 and Theorem 2 are polynomial time reductions when $\Sigma_m \cap \Sigma_{cA} = \emptyset$. \qed

Intuitively, if the actuator attacker cannot attack those unobservable events to the supervisor, then the reductions provided in Theorem 1 and Theorem 2, after removing the component $2^\ell$, are polynomial time reductions. In the following remark, we shall explain one implication of Theorem 3.

**Remark 1** If $\Sigma_m \cap \Sigma_{c} = \emptyset$, that is, when the given supervisor $S$ satisfies the normality property [19], then we also have $\Sigma_m \cap \Sigma_{cA} = \emptyset$, since $\Sigma_{cA} \subseteq \Sigma_{c}$. We consider this to be not quite restrictive. A supervisor satisfying the normality property leaves much reduced attack surfaces for the actuator attacker [12], since events that are disabled
are observable to the supervisor and enabling those disabled (attackable) events (by the actuator attacker) may immediately break the covertness of the attacker, if they are also enabled by the plant. Thus, being suspicious of the presence of an attacker, the designer may prefer to synthesize supervisors that satisfy the normality property. One option to realize the normality synthesis for the supervisors is by enlarging the observable event set \( \Sigma_o \) (e.g., deploy more sensors).

With the reductions provided in Theorem 1 and Theorem 2, the actuator attacker synthesis problem can be solved using the algorithm developed in [23], for both the damage-nonblocking goal and the damage-reachable goal. If there are some restrictions on the actuator attacker’s capability, then it is possible to reduce the synthesis complexity. For example, consider the case \( \Sigma_{c,A} \subseteq \Sigma_{o,A} \) where the normality synthesis procedure can be used [19], [24] for the synthesis of successful actuator attackers, we can use many off-the-shelf (symbolic) synthesis tools (see, for example, the tools STSLib [25] and Supremica [26]).

5 Discussions and Conclusions

In this work, we have presented a formulation of the actuator attacker synthesis problem in the formalism of discrete-event systems. In particular, we have addressed the problems of synthesis of covert actuator attackers with the damage-nonblocking goal and with the damage-reachable goal, respectively. The technique to solve the actuator attacker synthesis problem is by reduction to the well-studied supervisor synthesis problem, instead of developing new synthesis algorithms. In the rest of this section, we provide a discussion of some future works that can be carried out.

Synthesis of Powerful Actuator and Sensor Attackers: This work only considers the synthesis of actuator attackers. One of the immediate future works is to consider the synthesis of actuator and sensor attackers. In particular, we intend to address the synthesis of powerful attackers that are able to eavesdrop the control commands sent by the supervisor [12], [14].

The solution to the attacker synthesis problem has immediate applications to the resilient supervisor synthesis problem, as explained in [14].

Optimal Attraction Attacker: The attacker considered in this work only ensures damage-reachability or damage-nonblockingness. There is no mechanism for ensuring “necessity” of damage-infliction or optimization of attacker to achieve damage-infliction as soon as possible. This will be addressed in the future work.

Resilient Supervisor Synthesis: The problem of resilient supervisor synthesis needs to be addressed as well. In [14], a bounded resilient supervisor synthesis approach has been developed as a semi-decision procedure for the synthesis of resilient supervisors against powerful actuator attackers. One of the immediate future works is to consider the synthesis of bounded resilient supervisors against powerful actuator and sensor attackers. It is also of interest for us to explore other (semi-)decision procedures for resilient supervisor synthesis.

Arguably, the damage-nonblocking goal ensures “necessity” of damage-infliction in the long run.
Acknowledgements  This work is financially supported by Singapore Ministry of Education Academic Research Grant RG91/18-(S)-SU RONG (VP), which is gratefully acknowledged.

References

1. L. K. Carvalho, Y. C. Wu, R. Kwong, S. Lafortune, “Detection and prevention of actuator enablement attacks in supervisory control systems”, International Workshop on Discrete Event Systems, pp. 298-305, 2016.
2. L. K. Carvalho, Y. C. Wu, R. Kwong, S. Lafortune, “Detection and mitigation of classes of attacks in supervisory control systems”, Automatica, vol. 97, pp. 121-133, 2018.
3. R. Su, “Supervisor synthesis to thwart cyber-attack with bounded sensor reading alterations”, Automatica, vol. 94, pp. 35-44, 2018.
4. R. M. Goes, E. Kang, R. Kwong, S. Lafortune, “Stealthy deception attacks for cyber-physical systems”, Conference on Decision and Control, pp: 4224-4230, 2017.
5. R. Lanotte, M. Merro, R. Muradore, L. Viganò, “A formal approach to cyber-physical attacks”, IEEE Computer Security Foundations Symposium, pp: 436-450, 2017.
6. A. Jones, Z. Kong, C. Belton, “Anomaly detection in cyber-physical systems: A formal methods approach”, Conference on Decision and Control, pp: 848-853, 2014.
7. M. Wakaiki, P. Tabuada, J. P. Hespanha, “Supervisory control of discrete-event systems under attacks”, arXiv:1701.00881v1, 2017.
8. P. M. Lima, M. V. S. Alves, L. K. Carvalho, M. V. Moreira, “Security against network attacks in supervisory control systems”, IFAC, 50(1): 12333-12338, 2017.
9. P. M. Lima, L. K. Carvalho, M. V. Moreira, “Detectable and undetectable network attack security of cyber-physical systems”, IFAC, 51(7): 179-185, 2018.
10. E. Kang, S. Adepu, A. Mathur, D. Jackson, “Model based security analysis of a water treatment system”, SEsCPS, pp: 22-28, 2016.
11. M. Rocchietto, N.O. Tippenhauer, “Towards formal security analysis of industrial control systems”, ACM Asia Conference on Computer and Communications Security, pp: 114-126, 2017.
12. L. Lin, S. Thuijsman, Y. Zhu, S. Ware, R. Su, M. Reniers, “Synthesis of successful normal actuator attackers on normal supervisors”, American Control Conference, accepted, 2019.
13. Y. Zhu, L. Lin, R. Su, “Supervisor obfuscation against actuator enablement attack”, European Control Conference, accepted, 2018.
14. L. Lin, Y. Zhu, R. Su, “Towards bounded synthesis of resilient supervisors against actuator attacks”, [arXiv:1903.08358] 2019.
15. Y. Wang, A. K. Bozkurt, M. Pajic, “Attack-resilient supervisory control of discrete-event systems”, [arXiv:1904.03264] 2019.
16. X. Yin, S. Li, “Verification of opacity in networked supervisory control systems with insecure control channels”, Conference on Decision and Control, pp: 4851-4856, 2018.
17. Loïc Hélouët, Hervé Marchand, Laurie Ricker, “Opacity with powerful attackers”, 51(7): 464-471, 2018.
18. A. Rashidinejad, L. Lin, B. H. J. Watzele, Y. Zhu, M. Reniers, Rong Su, “Supervisory control of discrete-event systems under attacks: an overview and outlook”, European Control Conference, accepted, 2019.
19. W. M. Wonham, K. Cai, Supervisory Control of Discrete-Event Systems, Monograph Series Communications and Control Engineering, Springer, 2018.
20. C. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, Boston, MA: Kluwer, 1999.
21. J. E. Hopcroft, J. D. Ullman, Introduction to automata theory, languages, and computation, Addison-Wesley, Reading, Massachusetts, 1979.
22. A. Bergeron, “A unified approach to control problems in discrete event processes”, RAIRO-Theoretical Informatics and Applications, 27(6): 555-573, 1993.
23. X. Yin, S. Lafortune, “Synthesis of maximally permissive supervisors for partially observed discrete event systems”, IEEE Transactions on Automatic Control, 61(5):1239-1254, 2016.
24. D. Wang, L. Lin, Z. Li, W. M. Wonham, “State-based control of discrete-event systems under partial observation”, IEEE Access, 6(1): 42084-42093, 2018.
25. C. Ma, W. M. Wonham, “STSLib and its application to two benchmarks”, International Workshop on Discrete Event Systems, pp. 119-124, 2008.
26. R. Malik, K. Akeson, H. Flordal, M. Fabian, “Supremica-an efficient tool for large-scale discrete event systems”, IFAC, 50(1): 5794-5799, 2017.