Supervisory Control of Discrete Event Systems in the Presence of Sensor and Actuator Attacks

Yu Wang and Miroslav Pajic

***Abstract*— This work focuses on control of discrete event systems (DES) in the presence of attacks on their inputs and outputs. We propose to model such attacks as nondeterministic finite state transducers (FSTs) and show how FSTs can be used to capture a very wide class of attacks including all previously considered attacks on DES, as well as additional attacks and attack features reported in recent security incidents. We study the supervisory control problem in cases when attacks occur: (i) only on the sensors, (ii) only on the actuators, and (iii) both on the actuators and sensors of the plant. For each case, we present new sets of controllability theorems and synthesizing algorithms for attack-resilient supervisors. On a series of examples, we illustrate the use of our approach for modeling and design of such security-aware supervisory control.**

I. Introduction

Control systems operate in a range of security-critical domains where communication between the controller and the plant, or the physical environment of the plant, may be susceptible to adversarial attacks. Thus, it is critical to provide techniques to analyze behavior of control systems under attack, as well as synthesize attack-resilient control systems [1], [2], [3]. Such systems should be able to provide strong control guarantees under different types of intelligent and coordinated attacks (e.g., [1], [3]); such malicious behaviors are more complex than random failures, which is well studied in the fields of reliability and fault tolerant control.

In this work, we adopt the framework of the supervisory control of discrete event systems [4], in which a supervisor controls the behavior of a plant, modeled as a discrete event system (DES), within a desired language. Recent works focused on securing DES, consider possible attacks on communication between the supervisor and the plant [5], [7], [8], [9]; specifically, the attackers have the ability to insert, delete, and replace events/symbols communicated with the plant according to certain prespecified rules, and the supervisor, generally realized by a finite state automaton, regulates the words passing-by to counter the attacks.

We show that all previously studied attacks on DES including injection, deletion, and replacing events/symbols according to certain prespecified rules, fall into a more general class of regularly-rewriting attacks, which are considered in this work. With these attacks, attackers can revise the symbols communicated between the supervisor and the plant ***nondeterministically*** to a word (possibly empty) in a regular language. Mathematically, the regularly-rewriting attacks define a regular relation between the input and output languages of the attacker [10], [11]. We show how they can be realized by finite state transducers (FST) or nondeterministic Mealy machines. To counter such attacks and improve resiliency guarantees, we also consider supervisors that can be modeled as FSTs, instead of automata as previously done; this allows the supervisor to rewrite symbols, in addition to monitoring.

Figure 1 illustrates the considered supervisory control setup where FSTs model attacks on information delivered to the controller from the plant's sensors as well as attacks on control commands delivered to the actuators. We show how this configurations captures the basic controllability problems in attack-resilient supervisory control. Specifically, we focus on configurations with attacks on: (i) sensors, (ii) actuators, and (iii) both actuators and sensors. This enables capturing of network-based attacks that may corrupt communicated data (e.g., as in [18], [1]), as well as noninvasive attacks that affect the plant's environment (e.g., GPS spoofing attacks [19]).

We show that for Case (i), attacks on sensing data delivered to the controller can be completely countered by an FST-based supervisor derived by the serial composition of the inversion of the (very-general) attack model and a model of the desired language. For Case (ii), the actuator attacks on commands sent to the plant can be partly countered by a supervisor derived by the serial composition of a model of the desired language and the inversion of the attack model. The exact controllability is achieved if the desired language is invariant under the attack. Finally, for Case (iii), we show that an attack-resilient supervisor can be derived by serially composing the supervisors from (i) and (ii).

The derived controllability conditions for actuator attacks generalizes the standard controllability conditions for DESs under partial controllable and observable sets [4]. Our controllability conditions distinguish from existing works on the attack-resilient supervisory control of DESs [5], [7], [8], [9], as more general attack and supervisor models are adopted.

This paper is organized as follows. After preliminaries in Section II, we show how attacks can be modeled by FSTs (Section III) and formally specify the problem (Section IV). Controllability theorems and algorithms to design attack-resilient supervisors are presented in Section V and VI, when only plant sensors or actuators, respectively, are corrupted. These results are generalized for attacks on both sensors and actuators (Section VII), before concluding in Section VIII.

II. Preliminaries

The empty string is denoted by ε. A finite-length sequence of symbols taken from a given finite set is called a *word*. A set of words is called a *language* of the symbols. The cardinality and the power set of a set $\mathbf{I}$ are denoted by $|\mathbf{I}|$ and $2^{\mathbf{I}}$, respectively. For two sets $\mathbf{I}$ and $\mathbf{O}$, let $\mathbf{I}\backslash\mathbf{O} = \{\mathrm{i} \in \mathbf{I} \mid$

The authors are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27707, USA. Email:{yu.wang094, miroslav.pajic}@duke.edu

This work is sponsored by the awards ONR N00014-17-1-2012 and N00014-17-1-2504, AFOSR FA9550-19-1-0169, and NSF CNS-1652544.

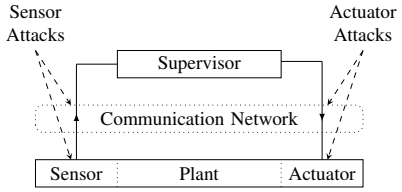


Fig. 1: Supervisory control in the presence of attacks on sensors, actuators as well as the communication between the sensors, controller and actuators.

$\mathsf{i} \notin \mathbf{O}\}$. For $n \in \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers, let $[n] = \{1, .., n\}$. For a word $I = \mathsf{i}_1\mathsf{i}_2 \dots \mathsf{i}_n$, we call $\mathsf{i}_1\mathsf{i}_2 \dots \mathsf{i}_k$, with $k \leq n$, a prefix of I. For a language L, its prefix-closure is defined by $\overline{L} = \{I \mid I \text{ is a prefix of } J, J \in L\}$. The language L is *prefix-closed* if $L = \overline{L}$. Also, we adopt the following convention on generating regular expressions: a superscript * means repeating a symbol or a set of symbols finitely many times, and a comma means "or".

A *relation* $\mathcal{R}$ between two sets $\mathbf{I}$ and $\mathbf{O}$ is a set $\mathcal{R} \subseteq \mathbf{I} \times \mathbf{O}$. For $\mathsf{i} \in \mathbf{I}$, let $\mathcal{R}(\mathsf{i}) = \mathcal{R}(\mathsf{i}, \cdot) = \{\mathsf{o} \in \mathbf{O} \mid (\mathsf{i}, \mathsf{o}) \in \mathcal{R}\}$. The relation $\mathcal{R}(\mathsf{i})$ is a *partial function* for the input i if $|\mathcal{R}(\mathsf{i})| \leq 1$, for any $\mathsf{i} \in \mathbf{I}$. More generally, for $\mathbf{I}' \subseteq \mathbf{I}$, while slightly abusing the notation we define $\mathcal{R}(\mathbf{I}') = \mathcal{R}(\mathbf{I}', \cdot) = \{\mathsf{o} \in \mathbf{O} \mid (\mathsf{i}', \mathsf{o}) \in \mathcal{R}, \mathsf{i}' \in \mathbf{I}'\}$. Thus, $\mathcal{R}(\cdot)$ defines a function $2^{\mathbf{I}} \to 2^{\mathbf{O}}$. For relation $\mathcal{R} \subseteq \mathbf{I} \times \mathbf{O}$, its inversion is defined by $\mathcal{R}^{-1} = \{(\mathsf{o}, \mathsf{i}) \in \mathbf{O} \times \mathbf{I} \mid (\mathsf{i}, \mathsf{o}) \in \mathcal{R}\}$. Finally, for two relations $\mathcal{R} \subseteq \mathbf{I} \times \mathbf{O}$ and $\mathcal{R}' \subseteq \mathbf{I}' \times \mathbf{O}'$, their (serial) composition is defined by $\mathcal{R} \circ \mathcal{R}' = \{(\mathsf{i}, \mathsf{o}') \in \mathbf{I} \times \mathbf{O}' \mid \exists \mathsf{o} \in \mathbf{O} \cap \mathbf{I}' : (\mathsf{i}, \mathsf{o}) \in \mathcal{R} \wedge (\mathsf{o}, \mathsf{o}') \in \mathcal{R}'\}$.

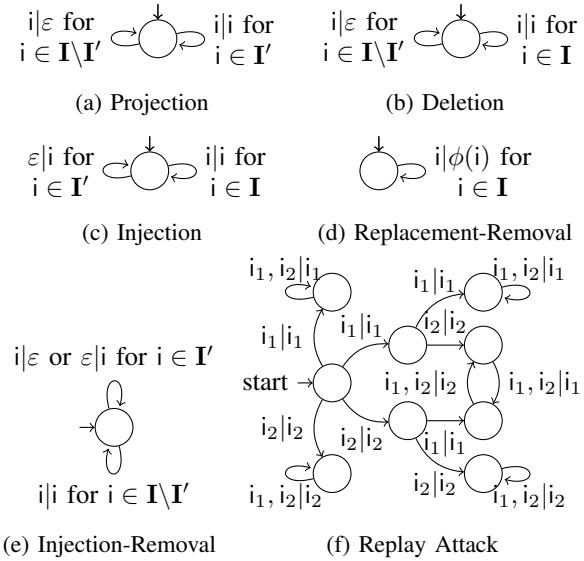
FSTs extend Finite State Automata (FSA) by generating an output sequence nondeterministically during execution, by augmenting each transition with a regular output language.

Definition 1 (Finite State Transducer)**.** *An FST is a tuple* $\mathcal{A} = (\mathsf{S}, \mathsf{s}_{\text{init}}, \mathbf{I}, \mathbf{O}, \mathsf{Trans}, \mathsf{S}_{\text{final}})$ *where: (i)* S *is a finite set of* states*; (ii)* $\mathsf{s}_{\text{init}} \in \mathsf{S}$ *is the* initial state*; (iii)* $\mathbf{I}$ *is a finite set of* inputs*; (iv)* $\mathbf{O}$ *is a finite set of* outputs*; (v)* $\mathsf{Trans} : \mathsf{S} \times \mathbf{I} \cup \{\varepsilon\} \to \mathbf{O} \cup \{\varepsilon\} \times \mathsf{S}$ *is a partial* transition *relation; and (vi)* $\mathsf{S}_{\text{final}} \subseteq \mathsf{S}$ *is a finite set of* final states*.*

The FST is deterministic if Trans is a partial function. The sequence $(\mathsf{s}_{\text{init}}, \mathsf{i}_0, \mathsf{o}_0, \mathsf{s}_0)(\mathsf{s}_0, \mathsf{i}_1, \mathsf{o}_1, \mathsf{s}_1) \dots (\mathsf{s}_{n-2}, \mathsf{i}_{n-1}, \mathsf{o}_{n-1}, \mathsf{s}_{n-1})$ is called an *execution*, if $(\mathsf{o}_i\mathsf{s}_i) \subseteq \mathsf{Trans}(\mathsf{s}_{i-1}, \mathsf{i}_i)$ for $i \in [n]$ with $\mathsf{s}_{-1} = \mathsf{s}_{\text{init}}$. The state s_{n-1} is called reachable upon receiving the input word $I = \mathsf{i}_0\mathsf{i}_1 \dots \mathsf{i}_n$. An input/output word is *accepted* by $\mathcal{A}$, if there exists such an execution ending at $\mathsf{S}_{\text{final}}$. The set of accepted input/output words is called the *input/output languages* of $\mathcal{A}$, denoted by $L_I(\mathcal{A})$ and $L_O(\mathcal{A})$, respectively.

We refer to [14] for common operations on FSTs, such as inversion and composition. Specifically, the complexity is $O(n_1)$ for inversion, and $O(n_1 n_2)$ for composition, where n_1 and n_2 are the numbers of transitions. Obviously, an FST $\mathcal{A}$ defines a regular relation $\mathcal{R}_{\mathcal{A}}$ between the input and output languages. On the other hand, a relation $\mathcal{R} \subseteq \mathbf{I}^* \times \mathbf{O}^*$ is *regular*, only if it is realized by an FST.

Remark 1. *In this work, FSA are viewed as FSTs with identical inputs and outputs. On the other hand, FSTs can be viewed as FSA with labels in* $(\mathbf{I} \cup \{\varepsilon\}) \times (\mathbf{O} \cup \{\varepsilon\})$.



(a) Projection (b) Deletion (c) Injection (d) Replacement-Removal (e) Injection-Removal (f) Replay Attack

Fig. 2: FST realizations of different attack models.

III. Modeling Attacks with FSTs

This section presents modeling attacks on supervisory control using FSTs; we show how such FST-based models generalize all existing attack models. Specifically, we start by showing how FSTs can be used to capture all previously reported attacks on DES [5], [9], as well as additional attacks and attack features. For example, FSTs can be used to capture constraints imposed by the system design, as well as model finite-memory replay attacks, where the attacker records a finite-length of symbols and replays it repeatedly [20], [2].

Example 1 (Projection/Deletion/Injection Attacks)**.** *Consider* $\mathbf{I}'$ *where* $\mathbf{I}' \subseteq \mathbf{I}$*. The projection attack defined as*

$$\text{Project}_{\mathbf{I}'}(\mathsf{i}) = \begin{cases} \mathsf{i}, & \text{if} \quad \mathsf{i} \in \mathbf{I}' \\ \varepsilon, & \text{otherwise,} \end{cases} \tag{1}$$

captures attacks that remove all symbols from $\mathbf{I} \setminus \mathbf{I}'$*. On the other hand, the (**nondeterministic**) deletion attack defined as*

$$\text{Delete}_{\mathbf{I}'}(\mathsf{i}) = \begin{cases} \mathsf{i}, & \text{if} \quad \mathsf{i} \in \mathbf{I}' \\ \varepsilon \text{ or } i, & \text{otherwise,} \end{cases} \tag{2}$$

extends the $\text{Project}_{\mathbf{I}'}$ *attack as it captures that the attacker may (or may not) remove symbols from* $\mathbf{I} \setminus \mathbf{I}'$*; e.g., if* $\mathbf{I}' = \mathbf{I}$ *this model captures Denial-of-Service (DoS) attacks [21].*

Finally, the (nondeterministic) injection attack defined as

$$\text{Insert}_{\mathbf{I}'}(\mathsf{i}) = (\mathbf{I}')^*\mathsf{i}(\mathbf{I}')^* \tag{3}$$

captures that a finite number of symbols from $\mathbf{I}'$ *can be added before and/or after the symbols. These attacks can be modeled as FSTs (Fig. 2a, 2b and 2c, respectively).* $\lhd$

Example 2 (Replacement-removal Attack)**.** *This attack is defined by the replacing-removing rule* $\phi : \mathbf{I} \to 2^{\mathbf{I} \cup \{\varepsilon\}}$*, and can be represented by an FST as shown in Figure 2d.* $\lhd$

Example 3 (Injection-removal Attack)**.** *Let* $\mathbf{I}' \subseteq \mathbf{I}$*. An injection-removal attack nondeterministically inserts or removes symbols in* $\mathbf{I}'$ *from a word (the FST in Fig. 2e).* $\lhd$

Example 4 (Finite-Memory Replay Attack)**.** *For systems with continuous-state dynamics, replay attacks have been modeled and studied (e.g., [20], [2]). On the other hand, for DESs no such models currently exit. In a DES, a replay attack records a prefix of a word and replaces the rest with*

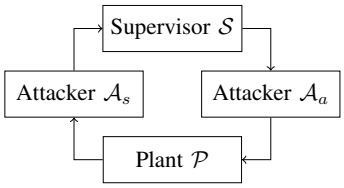


Fig. 3: Controllability under actuator and sensor attacks.

the repetitions of the recorded prefix, with the prefix size being bounded by the finite-memory capacity (i.e., size) N. For example, a replay attack recording a prefix of length up to $N = 2$ for any word of symbols $\mathbf{I} = \{\mathsf{i}_1, \mathsf{i}_2\}$ can be modeled by an FST as shown in Figure 2f. ◁

Remark 2. *Using FSTs to model attacks naturally supports the composition of multiple attacks modeled with the corresponding FSTs. With the general architecture from Figure 1, the system may be under a coordinated attack from multiple deployed attackers, capturing different 'point-of-entries' for the attack vectors on sensors/actuators and communication network; for example, false data injection via sensor spoofing on a subset of plant sensors (e.g., [19]) in coordination with DoS attacks on transmitted measurements from other sensors.*

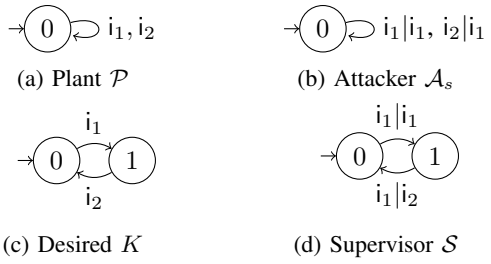
IV. Modeling and Problem Formulation

In this section, we introduce a general mathematical framework for the supervisory control of DESs subject to attacks on both the plant's actuators and sensors. As illustrated in Figure 3, the supervisor $\mathcal{S}$ controls behavior of the plant $\mathcal{P}$ by observing the symbols that $\mathcal{P}$ generates and then sending the possible control symbols back to it. Here, $\mathcal{P}$ is a DES driven by a finite set of symbols $\mathbf{I}$. Its state transits upon receiving an acceptable word $i \in \mathbf{I}^*$.

However, the information about the sensed symbols can be compromised by an attacker with the ability of inject, delete or replace symbols in both the control and observation. These attacks are modeled by two attack FSTs $\mathcal{A}_a$ and $\mathcal{A}_s$ on the actuators and sensors respectively, with the same sets of input and output symbols $\mathbf{I} = \mathbf{O}$. They can *regularly rewrite* an acceptable input word, i.e., replace a symbol *nondeterministically* with an arbitrary word taken from some predefined regular language. This includes, e.g., injection, replacement and deletion. ***We do not assume to know what actions the attacker may perform***. Rather, the FSTs use nondeterminism to capture all possible actions of the attacker for a specific set of compromised resources (e.g., sensors, actuators), as well as all potential limitations imposed on the attacker's actions by the system design (e.g., the use of cryptographic primitives on some communication messages to prevent false-data inserting attacks over the network).

Furthermore, in our setup the supervisor is also modeled by an FST $\mathcal{S}$. Using FSTs instead of automata to model supervisors provides them the ability to revise symbols that are required to counter attacks on the plant's sensors; Example 5 presents an attack $\mathcal{A}_s$ that can only be handled by FSTs.

Example 5. *Consider a set of symbols $\mathbf{I} = \{\mathsf{i}_1, \mathsf{i}_2\}$ and a plant $\mathcal{P}$ accepting the language $(\mathsf{i}_1, \mathsf{i}_2)^*$, as in Figure 4a. The (prefix-closed) desired language is $K = \overline{(\mathsf{i}_1\mathsf{i}_2)^*}$, represented by a discrete event system $\mathcal{M}_K$ in Figure 4c. The sensor attacks $\mathcal{A}_s$ are modeled by an FST as shown in Figure 4b.*



(a) Plant $\mathcal{P}$ (b) Attacker $\mathcal{A}_s$

(c) Desired K (d) Supervisor $\mathcal{S}$

Fig. 4: Sensor attacks that can only be countered by FSTs.

Supervising the plant to the desired language K without the ability to revise symbols is not feasible. The reason is that the supervisor output should be the desired language $(\mathsf{i}_1, \mathsf{i}_2)^*$, but the input is always i_1^*, since the plant only generates i_2^* and the attacker rewrites it to i_1^*. However, for such attack model there exists an FST-based attack-resilient supervisor $\mathcal{S}$ for the plant, shown in Figure 4d – it counters the attacks by revising i_1 back to i_2 every other step. ◁

We assume that the attack FSTs $\mathcal{A}_a$, $\mathcal{A}_s$, supervisor $\mathcal{S}$ and plant $\mathcal{P}$ only receive acceptable inputs (see Assumptions 1 and 2). This is generally achievable with the proper use of FST models for the attacks $\mathcal{A}_a$, $\mathcal{A}_s$, and supervisor $\mathcal{S}$.

For the described system, we consider the problem of attack-resilient supervisory control. Specifically, let $L(\mathcal{P})$ be the language generated by the plant without supervisor and $K \subseteq L(\mathcal{P})$ be the desired language. We consider the existential and, more importantly, the synthesis problem of a supervisor $\mathcal{S}$ supervising the language received by the plant to be exactly or as close as possible to some desired language K when the system is under attack. This is formally specified in Definition 2. It is important to note that the supervisor is not trying to completely recover a corrupted control; instead, it constrains the control sent to the plant to be within K.

To simplify our presentation, we assume that for the plant $\mathcal{P}$, supervisor $\mathcal{S}$, and actuator and sensor attacks $\mathcal{A}_a, \mathcal{A}_s$, all states are final $\mathsf{S}_{\text{final}} = \mathsf{S}$, i.e., both the sets of their inputs and outputs are prefix-closed. We also assume that the desired language is also prefix-closed $K = \bar{K}$ and regular. The regularity of K is to ensure that it is controllable by supervisors modeled by FSTs. The prefix-closeness requires that the supervision can be implemented step-by-step.

Definition 2. *The supervisor $\mathcal{S}$ weakly controls the deterministic plant $\mathcal{P}$ to the desired language K in the presence of actuator and sensor attacks $\mathcal{A}_a$ and $\mathcal{A}_s$, if it holds that*

$$K \subseteq_{\min} L(\mathcal{P}|\mathcal{S}, \mathcal{A}_a, \mathcal{A}_s). \tag{4}$$

Here, $L(\mathcal{P}|\mathcal{S}, \mathcal{A}_a, \mathcal{A}_s)$ denotes the restricted language of $\mathcal{P}$ under supervision with both actuator and sensor attacks, while $\subseteq_{\min}$ stands for minimal inclusion – i.e., any supervisor $\mathcal{S}'$ with $K \subseteq L(\mathcal{P}|\mathcal{S}', \mathcal{A}_a, \mathcal{A}_s)$ satisfies $L(\mathcal{P}|\mathcal{S}, \mathcal{A}_a, \mathcal{A}_s) \subseteq L(\mathcal{P}|\mathcal{S}', \mathcal{A}_a, \mathcal{A}_s)$. Furthermore, we say that the supervisor controls the plant $\mathcal{P}$ to the desired language K when the equality in (4) *holds.*

The restricted languages of $\mathcal{P}$ with only actuator or sensor attacks are denoted by $L(\mathcal{P}|\mathcal{S}, \mathcal{A}_a)$ and $L(\mathcal{P}|\mathcal{S}, \mathcal{A}_s)$, respectively. Both controllability and weak controllability for these cases are similarly defined.

Relationship with existing work: By specifying suitable actuator and sensor attack models, existing DES under at-

tack problem formulations can be derived from the general problem formulation considered in this work. To show this, let the actuator attack model $\mathcal{A}_a^{(s)} = \text{Insert}_{\mathbf{I}_{uc}} \circ \mathcal{P}$ be the serial composition of the injection attack from (3) and the plant. These inserted symbols are acceptable symbols of the plant, but not uncontrollable by the supervisor. In addition, let the sensor attack model $\mathcal{A}_s^{(s)} = \text{Project}_{\mathbf{I}_o}$ from (1). The removed symbols can be viewed as the unobservable symbols generated by the plant. For such setup, our problem reduces to the standard (i.e., without taking security/attacks into account) supervisory control formulation [4] with uncontrollable events $\mathbf{I}_{uc}$, and unobservable events $\mathbf{I}_{uo} = \mathbf{I} \backslash \mathbf{I}_o$.

In the second setup, let us assume that actuator and sensor attack modules $\mathcal{A}_a^{(s)}$ and $\mathcal{A}_s^{(s)}$ (defined above) are composed with modules of injection-removal of a set of vulnerable control symbols from Example 3. Our problem for such setup directly captures the problem of supervisory control under the actuator and sensor enablement/disablement attack studied in [9]. Yet, if $\mathcal{A}_s^{(s)}$ is composed with a sensor replacement-removal attack from Example 2, the problem considered in this paper yields the problem of supervisory control under replacement-removal sensor attacks [5]. Similarly, composing $\mathcal{A}_s^{(s)}$ with a sensor injection-deletion attack from Example 3 maps our problem into supervisory control problem under injection-deletion sensor attacks from [5].

V. Controllability Under Attacks on Sensors

We start our analysis with the supervisory control problem under only sensor attacks. In the rest of the section, we make the following assumption on the sensor attacker $\mathcal{A}_s$.

Assumption 1. *The attack FST $\mathcal{A}_s$ can (i) accept and (ii) only accept words that are acceptable to the plant $\mathcal{P}$, i.e., $L_I(\mathcal{A}_s) = L(\mathcal{P})$.*

The first part of Assumption 1 means that the attack FST $\mathcal{A}_s$ is well-defined for any acceptable word of the plant $\mathcal{P}$. This can be done by encoding the error behavior of receiving an unacceptable word to the plant model. The second part of Assumption 1 is always achievable by trimming the $\mathcal{A}_s$.

To counter the attack modeled by $\mathcal{A}_s$, we can construct its inversion $\mathcal{A}_s^{-1}$. For any word k passing the plant, the attack $\mathcal{A}_s$ rewrites it to a word in $\mathcal{R}_{\mathcal{A}_s}(k)$, and the inversion can reverse it back to $\mathcal{R}_{\mathcal{A}_s \circ \mathcal{A}_s^{-1}}(k)$; note that $\mathcal{R}_{\mathcal{A}_s \circ \mathcal{A}_s^{-1}}(k)$ is the set of possible words passing through the plant and yielding the same observation after attack as k. Restricting this set of words to the desired language K guarantees the supervisory control goal. Therefore, the supervisor $\mathcal{S}$ should be designed to be $\mathcal{A}_s^{-1} \circ \mathcal{M}_K$, where $\mathcal{M}_K$ is the automata realizing the desired language K. In this way, it is guaranteed that the supervisor only sends control words within K, even if the sensors are corrupted, as summarized by the theorem below.

Theorem 1 (Controllability under sensor attacks)**.** *The plant $\mathcal{P}$ is controllable to the desired regular language $K \subseteq L(\mathcal{P})$ under attack $\mathcal{A}_s$ on the plant's sensors. This can be achieved by the supervisor $\mathcal{S} = \mathcal{A}_s^{-1} \circ \mathcal{M}_K$.*

Furthermore, Theorem 1 also directly provides a computational method to design such attack-resilient supervisor; the procedure is introduced in Algorithm 1.

Example 6 (Supervisor design under sensor attacks)**.** *From Example 5, the supervisor $\mathcal{S} = \mathcal{A}_s^{-1} \circ \mathcal{M}_K$ in Figure 4d is derived by a serial composition of the inversion of $\mathcal{A}_s$ in Figure 4b and the model of desired language K (Figure 4c).* $\triangleleft$

Algorithm 1 Design Supervisor Resilient to Sensor Attacks

Require: Desire language K, plant $\mathcal{P}$, sensor attacker $\mathcal{A}_s$.
1: Find $\mathcal{M}_K$ realizing $K \subseteq L(\mathcal{P})$.
2: Compute inversion $\mathcal{A}_s^{-1}$.
3: Compute serial composition $\mathcal{S} = \mathcal{A}_s^{-1} \circ \mathcal{M}_K$.
4: **return** Supervisor $\mathcal{S}$.

Remark 3. *By Theorem 1, attacks on sensors do not affect controllability, as opposed to previous works [5], [7], [9]. This is caused by the fact that the plant is deterministic, thus the FST-based supervisor can control the plant in open-loop without using the sensing information of the plant. These sensing information will become useful to learn the state of the plant when it is nondeterministic, as we show in [6].*

VI. Controllability Under Actuator Attacks

We now study the attack-resilient supervisory control problem when only attacks on plant actuators may occur. Specifically, actuators of the plant $\mathcal{P}$ are under the attack $\mathcal{A}_a$, but the supervisor $\mathcal{S}$ has direct access to the words passing the plant. This problem is more complex then the supervisory control problem with sensor attacks studied in Section V, as the supervisor commands are not directly sent to the plant. Consequently, not all desired language are controllable to the plant. For example, if attack $\mathcal{A}_a$ generates the empty word ε upon all inputs, modeling DoS attacks, then the only controllable desired language for the plant is $\{\varepsilon\}$.

We make the following assumption on the attack FST $\mathcal{A}_a$.

Assumption 2. *The attack FST $\mathcal{A}_a$ can (i) generate and (ii) only generate words that are acceptable to the plant $\mathcal{P}$, i.e., $L_O(\mathcal{A}_a) = L(\mathcal{P})$.*

The first part of Assumption 2 means that the actuator attack $\mathcal{A}_a$ will not cause an error by sending an unacceptable input of the plant $\mathcal{P}$ (such attacks are easy to detect); instead, the attack will try to affect the plant and violate the control goal by sending a word in $L(\mathcal{P}) \backslash K$ to the plant. The second part ensures that every word in $L(\mathcal{P})$ may possibly be sent to the plant $\mathcal{P}$ – otherwise, we can only discuss controllability with respect to $L_O(\mathcal{A}_a)$ (see Remark 5).

For attack-resilient supervision of the plant under actuator attacks, the supervisor $\mathcal{S}$ needs to (i) constrain (i.e., filter) the words passing the plant $\mathcal{P}$, and (ii) rewrite these words to counter the effects of the actuator attacks $\mathcal{A}_a$.

Filter: Unlike our approach presented in Section V, for the first requirement, we construct an FST filter $\mathcal{M}_K$ of the desired language $K \in L(\mathcal{P})$. This is because an automaton of K cannot handle input words in $L(\mathcal{P}) \backslash K$. The filter takes a word $k \in L(\mathcal{P})$ and sends the same word if $k \in K$, and ε otherwise. This is achieved with Algorithm 2.

For the second requirement, we construct the inversion $\mathcal{A}_a^{-1}$ of the actuator attacks. Consequently, the supervisor is $\mathcal{S} = \mathcal{M}_K \circ \mathcal{A}_a^{-1}$. For any word k passing $\mathcal{M}_K$, we know $k \in K$. The inversion $\mathcal{A}_a^{-1}$ rewrites it to a word in $\mathcal{R}_{\mathcal{A}_a^{-1}}(k)$, and the actuator attack $\mathcal{A}_a$ rewrites it to $\mathcal{R}_{\mathcal{A}_a^{-1} \circ \mathcal{A}_a}(k)$ and passes to the plant. This is the set of possible words that can be derived by attacking the input words yielding k. Therefore, we have $k \in \mathcal{R}_{\mathcal{A}_a^{-1} \circ \mathcal{A}_a}(k)$ and $K \subseteq \mathcal{R}_{\mathcal{A}_a^{-1} \circ \mathcal{A}_a}(K)$.

Algorithm 2 FST Filter for Desired Language K.

Require: The desire language K, plant $\mathcal{P}$.
1: Find automata $\mathcal{M}$ and $\mathcal{M}'$ with $L(\mathcal{M}) = L(\mathcal{P})$ and $L(\mathcal{M}') = K$.
2: Convert $\mathcal{M}$ and $\mathcal{M}'$ to FSTs by adding ε as output and input symbol for each transition, respectively.
3: $\mathcal{A} = \mathcal{M} \circ \mathcal{M}'$.
4: Trim off transitions with input symbol ε in $\mathcal{A}$.
5: **return** Supervisor $\mathcal{A}$.

Algorithm 3 Design Supervisor under Actuator Attacks

Require: Desire language K, plant $\mathcal{P}$, actuator attacker $\mathcal{A}_a$.
1: Find FST filter $\mathcal{M}_K$ for K .
2: Compute inversion $\mathcal{A}_a^{-1}$.
3: Compute serial composition $\mathcal{S} = \mathcal{M}_K \circ \mathcal{A}_a^{-1}$.
4: **if** the output language $L_O(\mathcal{S} \circ \mathcal{A}_a) \subseteq K$ **then**
5: **return** K is controllable
6: **else**
7: **return** K is not controllable
8: **end if**
9: **return** Supervisor $\mathcal{S}$.

Since $\mathcal{R}_{\mathcal{A}_a^{-1} \circ \mathcal{A}_a}(K)$ is not necessarily contained in K, the supervisor $\mathcal{S} = \mathcal{M}_K \circ \mathcal{A}_a^{-1}$ only restricts the language passing the plant to a minimal superset of K. The desired language K is controllable, when the containment holds. This is equivalent to checking if K is contained in the output language of $\mathcal{M}_k \circ \mathcal{A}_a^{-1} \circ \mathcal{A}_a$. In this way, it is guaranteed that the plant only receives control words from K, even if the actuators are corrupted. This is summarized below.

Theorem 2 (Controllability under actuator attack)**.** *The plant $\mathcal{P}$ is weakly controllable to the desired regular language $K \subseteq L(\mathcal{P})$ under the attacks $\mathcal{A}_a$ on its actuators by the supervisor $\mathcal{S} = \mathcal{M}_k \circ \mathcal{A}_a^{-1}$. Accordingly, the minimal controllable language containing K is*

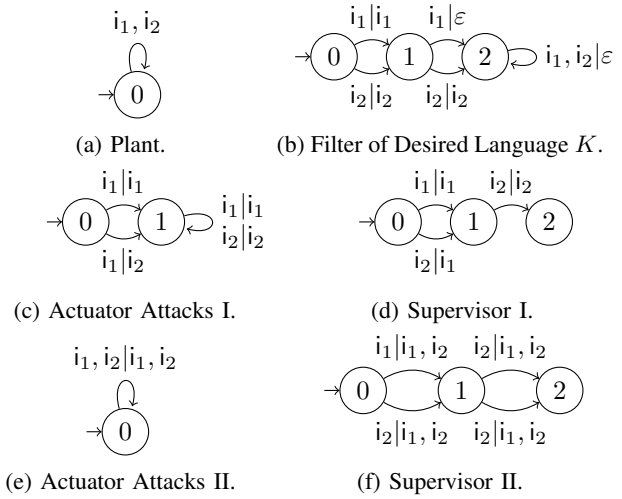
$$\tilde{K} = \mathcal{R}_{\mathcal{A}_a^{-1} \circ \mathcal{A}_a}(K). \tag{5}$$

The desired language is controllable if and only if $\tilde{K} = K$, or equivalently the output language of $\mathcal{M}_k \circ \mathcal{A}_a^{-1} \circ \mathcal{A}_a$ satisfies

$$L_O(\mathcal{M}_k \circ \mathcal{A}_a^{-1} \circ \mathcal{A}_a) \subseteq K. \tag{6}$$

Theorem 2 provides a computational method to design such supervisor, as provided in Algorithm 3.

Example 7 (Supervisor design for actuator attacks)**.** *For the setup from Figure 3 without attacks on sensors, let us consider a set of symbols $\mathbf{I} = \{\mathsf{i}_1, \mathsf{i}_2\}$ and a plant $\mathcal{P}$ accepting the language $(\mathsf{i}_1, \mathsf{i}_2)^*$, as shown in Figure 5a. The (prefix-closed) desired language is $K = \overline{(\mathsf{i}_1, \mathsf{i}_2)\mathsf{i}_2}$, associated with the FST filter $\mathcal{M}_K$ (Figure 5b). Now we consider the following two cases.* ***Controllable****: Attacks on actuators are represented by an FST $\mathcal{A}_a$ shown in Figure 5c. It rewrites the first i_1 nondeterministically to i_1 or i_2. The output language $L_O(\mathcal{S} \circ \mathcal{A}_a) = \overline{(\mathsf{i}_1, \mathsf{i}_2)\mathsf{i}_2}$ is exactly K. Therefore, the plant is controllable to K by the supervisor $\mathcal{S}$ even under the attack.* ***Weakly Controllable****: The actuator attacks $\mathcal{A}_a$ are represented by an FST from Figure 5e. It rewrites any i_1 nondeterministically to i_1 or i_2. It sends first i_1 and then i_1 or i_2 upon receiving i_2. The output language $L_O(\mathcal{S} \circ \mathcal{A}_a) = \overline{(\mathsf{i}_1, \mathsf{i}_2)(\mathsf{i}_1, \mathsf{i}_2)}$ is larger than K. Therefore, the plant is uncontrollable to K by the supervisor $\mathcal{S}$. It is easy to see that $\overline{(\mathsf{i}_1, \mathsf{i}_2)(\mathsf{i}_1, \mathsf{i}_2)}$ is a minimal superset of K.* ◁



(a) Plant. (b) Filter of Desired Language K.

(c) Actuator Attacks I. (d) Supervisor I.

(e) Actuator Attacks II. (f) Supervisor II.

Fig. 5: Example supervisors for actuator attacks.

Remark 4. *Note that the supervisor in Figure 5f, derived in Example 7, is nondeterministic — at the state 0, it can either send i_1 or i_2 upon receiving i_1. The controllability theorem guarantees that in the presence of attacks, the union of all possible words received by the plant under all these allowable controls is exactly the desired language K. In implementation, the nondeterminism can be resolved by choosing one of the allowable controls. Accordingly, the possible words received by the plant is contained in K.*

VII. Controllability Under Attacks on Both Actuators and Sensors

Finally, we study the setup with attacks on both plant actuators and sensors, as shown in Figure 3. This is a combination of theproblems studied in Sections V and VI. Again, we assume that the actuator attacks $\mathcal{A}_a$ and sensor attacks $\mathcal{A}_s$ are well-defined as stated in Assumptions 1 and 2.

From Section VI, it follows that the FST $\mathcal{M}_K \circ \mathcal{A}_a^{-1}$ can constrain the words passing the plant $\mathcal{P}$, and revise these words to counter the actuator attacks $\mathcal{A}_a$. This, in combination with the analysis in Section V, implies that the supervisor $\mathcal{A}_s^{-1} \circ \mathcal{M}_K \circ \mathcal{A}_a^{-1}$ can additionally counter the sensor attacks $\mathcal{A}_s$. It is easy to check that $K \subseteq \mathcal{R}_{\mathcal{A}_s \circ \mathcal{S} \circ \mathcal{A}_a}(K)$. However, $\mathcal{R}_{\mathcal{A}_s \circ \mathcal{S} \circ \mathcal{A}_a}(K)$ is not necessarily contained in K. The supervisor $\mathcal{S} = \mathcal{A}_s^{-1} \circ \mathcal{M}_K \circ \mathcal{A}_a^{-1}$ only restricts the language passing the plant to a minimal superset of K. The desired language K is controllable, when the containment holds. This is summarized by the following theorem.

Theorem 3 (Controllability under both attacks)**.** *For the system from Figure 3, the plant $\mathcal{P}$ is weakly controllable to the desired regular language $K \subseteq L(\mathcal{P})$ under the attacks $\mathcal{A}_a$ and $\mathcal{A}_s$ on its actuators and sensors, respectively, by the supervisor $\mathcal{S} = \mathcal{A}_s^{-1} \circ \mathcal{M}_K \circ \mathcal{A}_a^{-1}$. Accordingly, the minimal controllable language containing K is*

$$\tilde{K} = \mathcal{R}_{\mathcal{A}_a^{-1} \circ \mathcal{A}_a}(K). \tag{7}$$

The desired language is controllable if and only if $\tilde{K} = K$, or equivalently the output language of $\mathcal{M}_K \circ \mathcal{A}_a^{-1} \circ \mathcal{A}_a$ satisfies

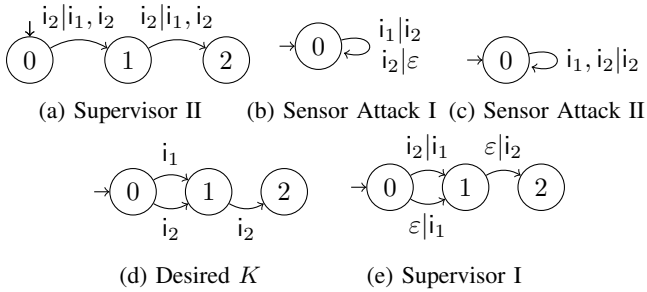
$$L_O(\mathcal{M}_K \circ \mathcal{A}_a^{-1} \circ \mathcal{A}_a) \subseteq K. \tag{8}$$

From Theorem 3, the design of an attack-resilient supervisor can be performed by separately taking into accounts

Algorithm 4 Supervisor Resilient to Actuator and Sensor Attacks

Require: Plant $\mathcal{P}$, actuator attacker $\mathcal{A}_a$, sensor attacker $\mathcal{A}_s$, Desire language K.

```
Find a model M_K of K .
Compute inversion A_a^{-1} and A_s^{-1} .
Compute serial composition S = A_s^{-1} ∘ M_K ∘ A_a^{-1} .
if the output language L_O(A_s ∘ S ∘ A_a) ⊆ K then
    return K is controllable
else
    return K is not controllable
end if
return Supervisor S.
```



(a) Supervisor II (b) Sensor Attack I (c) Sensor Attack II

(d) Desired K (e) Supervisor I

Fig. 6: Example supervisors for sensor and actuator attacks.

the actuator and sensor attacks, and the attacks on sensors $\mathcal{A}_s$ have no influence on the controllability. This result is different from most previous works [9], [4], [5], and is caused by the fact that the supervisor, when modeled by an FST, and not an automaton, has more power in generating control commands (i.e., can generate controls by itself); thus, it depends much less on the input word it receives. By adding a component $\mathcal{A}_s^{-1}$, the effect of the sensor attacker $\mathcal{A}_s$ is totally countered. Theorem 3 also provides a computational method to design such supervisor, as given in Algorithm 4.

Example 8 (Supervisor design for both sensor and actuator attacks)**.** *From Example 7, consider a set of symbols* $\mathbf{I} = \{\mathsf{i}_1, \mathsf{i}_2\}$ *and a plant* $\mathcal{P}$ *accepting the language* $(\mathsf{i}_1, \mathsf{i}_2)^*$*, as shown in Figure 5a. The (prefix-closed) desired language is* $K = \overline{(\mathsf{i}_1, \mathsf{i}_2)\mathsf{i}_2}$*, represented by an automaton* $\mathcal{M}_K$ *(Figure 6d). Now, let us consider the following two cases.* ***Controllable****: The attacks* $\mathcal{A}_a$ *and* $\mathcal{A}_s$ *are modeled by FSTs from Figures 5c and 6b, respectively. The* $\mathcal{A}_a$ *rewrites the first* i_1 *nondeterministically to* i_1 *or* i_2*. The* $\mathcal{A}_s$ *removes* i_2 *and replaces* i_1 *with* i_2*. The supervisor shown in Figure 6e does not contain any transition with input label* i_1*, as it will never appear due to the attack. The output language* $L_O(\mathcal{A}_s \circ \mathcal{S} \circ \mathcal{A}_a) = \overline{(\mathsf{i}_1, \mathsf{i}_2)\mathsf{i}_2}$ *is equal to* K*. Thus, the plant is controllable to* K *by the supervisor* $\mathcal{S}$*.* ***Weakly Controllable****: Consider attack FSTs* $\mathcal{A}_a$ *and* $\mathcal{A}_s$ *from Figures 5e and 6c, where* $\mathcal{A}_a$ *rewrites* i_1 *nondeterministically to* i_1 *or* i_2*, and* $\mathcal{A}_s$ *replaces* i_1 *with* i_2*. The supervisor does not contain any transition with input label* i_1*, as it will never appear. Hence, the output language* $\overline{(\mathsf{i}_1, \mathsf{i}_2)(\mathsf{i}_1, \mathsf{i}_2)}$ *minimally contains* K*. Comparing to Example 7, adding sensor attacks has no influence on controllability. This agrees with Remark 3.* $\lhd$

Finally, note that if the second part of Assumption 2 is violated, i.e., $L_O(\mathcal{A}_a) \subseteq L(\mathcal{P})$, Theorems 2 and 3 still hold on the trimmed plant accepting $L_O(\mathcal{A}_a)$.

Remark 5. *For* $L_O(\mathcal{A}_a) \subseteq L(\mathcal{P})$*,* $\tilde{K} = \mathcal{A}_a^{-1} \circ \mathcal{A}_a(K)$ *is the minimal controllable language containing* $K \cap L_O(\mathcal{A}_a)$*, and* K *is controllable if* $L_O(\mathcal{M}_K \circ \mathcal{A}_a^{-1} \circ \mathcal{A}_a) \subseteq K$ *and* $K \subseteq L_O(\mathcal{A}_a)$*.*

VIII. Conclusions

We have studied the attack-resilient supervisory control problem when attacks occur on: (i) sensors, (ii) actuators, and (iii) both actuators and sensors; we have considered a very general class of attacks and proposed to model them using FSTs. We have presented new sets of controllability conditions and synthesizing algorithms for supervisors in these scenarios. We have shown that for (i), an FST-based supervisor derived by the serial composition of the inversion of the attack model and a model of the desired language should be used; for (ii), the actuator attacks can be partly countered by a supervisor derived by the serial composition of a model of the desired language and the inversion of the attack model; and for (iii), a supervisor can be derived by serially composing the supervisors from (i) and (ii).

References

[1] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, "Attack models and scenarios for networked control systems," in *Conf. on High Confid. Net. Sys. (HiCoNS)*, 2012, pp. 55–64.

[2] Y. Mo, T.-H. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli, "Cyber–physical security of a smart grid infrastructure," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 195–209, 2012.

[3] M. Pajic, I. Lee, and G. J. Pappas, "Attack-Resilient State Estimation for Noisy Dynamical Systems," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 1, pp. 82–92, 2017.

[4] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York, NY: Springer, 2008.

[5] M. Wakaiki, P. Tabuada, and J. P. Hespanha, "Supervisory Control of Discrete-event Systems under Attacks," *arXiv:1701.00881*, 2017.

[6] Y. Wang, A. K. Bozkurt, and M. Pajic, "Attack-Resilient Supervisory Control of Discrete Event Systems," *arXiv:1904.03264*, 2019.

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

[8] R. M. Goes, E. Kang, R. Kwong, and S. Lafortune, "Stealthy deception attacks for cyber-physical systems," in *IEEE 56th Conference on Decision and Control (CDC)*, 2017, pp. 4224–4230.

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

[10] J. Sakarovitch and R. Thomas, "Elements of Automata Theory," p. 784, 2003.

[11] M. Droste, W. Kuich, and H. Vogler, Eds., *Handbook of Weighted Automata*, Springer-Verlag, 2009.

[12] M. Mohri, "Finite-State Transducers in Language and Speech Processing," *Computational Linguistics*, vol. 23, p. 42, 1997.

[13] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[14] M. Mohri, "Weighted Finite-State Transducer Algorithms. An Overview," in *Formal Languages and Applications*. 2004, pp.551–563.

[15] ——, "Weighted Automata Algorithms," in *Handbook of Weighted Automata*. Springer Berlin Heidelberg, 2009, pp. 213–254.

[16] C. Allauzen and M. Mohri, "Efficient algorithms for testing the twins property," p. 29, 2003.

[17] I. Jovanov and M. Pajic, "Relaxing integrity requirements for attack-resilient cyber-physical systems," *IEEE Transactions on Automatic Control*, pp. 1–1, 2019, to appear.

[18] R. Smith, "A decoupled feedback structure for covertly appropriating networked control systems," *IFAC World Congress*, pp. 90–95, 2011.

[19] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful gps spoofing attacks," in *18th ACM Conf. on Computer and Com. Security*, ser. CCS, 2011, pp. 75–86.

[20] F. Miao, M. Pajic, and G. Pappas, "Stochastic game approach for replay attack detection," in *IEEE 52nd Annual Conference on Decision and Control (CDC)*, Dec 2013, pp. 1854–1859.

[21] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *computer*, vol. 35, no. 10, pp. 54–62, 2002.