This article has been accepted for publication in IEEE Control Systems Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2022.3168926

On Resilient Supervisory Control Against Indefinite Actuator Attacks in Discrete-Event Systems

Ziyue Ma, *Member, IEEE*, Kai Cai, *Senior Member, IEEE*

***Abstract*— In this paper we study a resilient supervisory control design problem in discrete-event systems. Consider that there are certain unsafe states in the system that must be prevented from entering, and this can be ensured by a supervisor disabling certain controllable events. Also consider that the system is subject to actuator attacks from intruders: some controllable events disabled by a supervisor may be re-enabled by an intruder. Our purpose is to address a challenging scenario where the controllable events that are vulnerable to attacks are *indefinite*, i.e., any controllable event can be attacked. Associating to each unsafe state with a required *safety level* (a positive integer), our aim of this work is to design a resilient supervisor such that for every unsafe state q, if the number of actuator attacks is no greater than the safety level of q, then the controlled system is guaranteed to avoid entering q. We first encode the behavior of the system under attack into an automaton called the *resiliency automaton*. We then show that the resilient supervisor synthesis problem may be cast into a supervisory control problem in the resiliency automaton. Hence, a maximally permissive resilient supervisor can be obtained by using the Ramadge-Wonham supervisory control paradigm. To the best of our knowledge, this is the first result on supervisory control design against indefinite actuator attacks in discrete-event systems.**

***Index Terms*— Secret protection, security, discrete-event systems, automata**

I. Introduction

SAFE and resilient control against potential attacks in cyber-physical systems has drawn much attention in recent years [1]–[4], [6]–[9], [11], [15]. It is required that a plant be tolerant for potential attacks from external malicious intruders. The aim of an intruder is to lead the plant to unsafe or critical states by altering the information transmitted between the plant and the supervisor. Therefore, a supervisor must be resilient to guarantee the normal functionalities of a system under potential attacks.

In the last decade, researchers in discrete-event systems (DES) have done much work on the analysis and synthesis of resilient supervisors against various types of attacks. By *sensor attacks* [9], [11], [19], [20] an intruder can modify the information of sensor reading to cheat the supervisor to issue some wrong control actions. On the other hand, by *actuator attacks* [2], [8] an intruder can alter the control command from the supervisor to re-enable some events that are supposed to be disabled. Some works also consider the combined types of attacks [3], [7], [15]. The work in [6], [11] studies this problem from the dual point of view by solving a covert attacker synthesis problem. Besides, the work in [1], [3], [4] focus on the intrusion detection where the aim of the operator of a plant is to detect the invasion of an intruder by detecting abnormal behaviors in the plant. Moreover, passive/eavesdropper attacks are also studied (such as opacity [13], [5], [17], [10]) that are to infer some secrets in the system without interfering the behavior of it.

In this work we focus on the design of resilient supervisors against actuator attacks, i.e., some controllable events disabled by a supervisor may be re-enabled by an intruder so that they can be unexpectedly executed. In the work on actuator attacks in the literature (e.g., [2], [3], [7], [8], [15]), it is usually assumed that some events are not attackable by the intruder, while other events are vulnerable to attacks and can be attacked infinitely often. However, such an assumption need not be satisfied in practice. On one hand, we (the designer of the plant) often do not have *a priori* knowledge of which events may be attacked or of the capability of the attackers, which implies that it is unknown to us which events are vulnerable or not to attackers. In other words, the events that can be attacked by the intruder are *indefinite*. On the other hand, since an unexpected execution of a "disabled" event is an abnormal behavior, it should be immediately noticeable by the supervisor. This means that an intruder may not be able to perform actuator attacks infinitely often but only within a limited time window.

By the motivations above, in this paper we study the design of resilient supervisors against *indefinite actuator attacks* in DES. In the plant some states are considered to be unsafe and should be prevented from entering. Each unsafe state is associated with a required *safety level* depending on the importance of preventing the plant from entering it. Our aim is to design a resilient supervisor such that for every unsafe state q, if the number of actuator attacks is no greater than the safety level of q, then the controlled system is guaranteed to avoid entering q. Note that conventional framework of actuator

This work was supported in part by Shaanxi Provincial Natural Science Foundation under Grant No. 2022JM-323, the Fundamental Research Funds for the Central Universities under Grant JB210413, and JSPS KAKENHI Grant no. 21H04875 and the 2021 Osaka City University Strategic Research Grant to Encourage Applicants for Upper Level KAKENHI.

Z. Ma is with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (e-mail: maziyue@xidian.edu.cn).

K. Cai is with the Department of Core Informatics, Osaka Metropolitan University, Osaka 558-8585, Japan (e-mail: cai@omu.ac.jp).

This article has been accepted for publication in IEEE Control Systems Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2022.3168926

attacks is incomparable with the problem we consider, and the existing methods cannot be used here. In fact, since all events can be attacked, the methods in [2], [6], [8], [11] always return no solution. On the other hand, we do not consider sensor attacks [9], [11], [19], [20] on the observation channel (that is related to sensor readings), which implies that the plant is fully observable to the supervisor. Also, due to the limit of space, we assume that the plant is deterministic. However, our method can be straightforwardly generalized to partially observed, nondeterministic systems since the main results developed in this work also hold in such cases.

The main contents and contributions of this paper are summarized as follows. We first encode the behavior of the system under attack into an automaton called the *resiliency automaton*. Then, we prove that the resilient supervisor synthesis problem can be transformed into a standard supervisory control problem in the resiliency automaton. Hence, a maximally permissive resilient supervisor can be obtained by using the Ramadge-Wonham supervisory control paradigm. To the best of our knowledge, this is the first result on supervisory control design against indefinite actuator attacks in DES.

II. Preliminaries

A. Deterministic Finite Automaton

A *deterministic finite automaton* (*automaton* for short) is a four-tuple $G = (Q, \Sigma, \delta, q_0)$ where Q is a set of *states*; Σ is a set of *events*; $\delta : Q \times \Sigma \to Q$ is the partial transition function; and $q_0 \in Q$ is the initial state.

We use Σ^* to denote the *Kleene closure* of Σ, consisting of all finite sequences composed by the events in Σ (including the *empty sequence* ε). Given a sequence $s \in \Sigma^*$, $|s|$ denotes the *length* of s. The transition function δ is extended to $\delta^* : Q \times \Sigma^* \to Q$ by recursively defining $\delta^*(q, \varepsilon) = q$ and $\delta^*(q, s\sigma) = \delta(\delta^*(q, s), \sigma)$, where $s \in \Sigma^*$ and $\sigma \in \Sigma$. Henceforth for simplicity we write δ for δ^*. The *language* of G, denoted by $L(G)$, is defined as $L(G) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q\}$.

We use $\Gamma_G(q) = \{\sigma \in \Sigma \mid \delta(q, \sigma) \text{ is defined}\}$ to denote the set of events that can occur at state $q \in Q$, and we use $\Gamma_G(s) = \Gamma_G(\delta(q_0, s))$ to denote the set of events that can occur after sequence s.

Given an automaton $G = (Q, \Sigma, \delta, q_0)$, the *accessible part* of G, denoted as $Ac(G)$, is the automaton $G' = (Q', \Sigma, \delta', q_0)$ obtained from G by removing all unreachable states and their corresponding transitions. Precisely speaking, $Q' = \{q \in Q \mid (\exists s \in L(G))\ \delta(q_0, s) = q\}$, and δ' is the restriction of δ to $Q' \times \Sigma \to Q'$.

A sequence $\bar{s} \in \Sigma^*$ is a *prefix* of a sequence $s \in \Sigma^*$ if $s = \bar{s}s'$ where $s' \in \Sigma^*$. We use $\bar{s}_k$ (where $0 \le k \le |s|$) to denote the prefix of s of length k, i.e., $s = \bar{s}_k s'$ where $|\bar{s}_k| = k$ and $s' \in \Sigma^*$. The *prefix closure* of a language $L \subseteq \Sigma^*$ is the set $\overline{L} = \{s \in \Sigma^* \mid (\exists s' \in \Sigma^*) ss' \in L\}$.

B. Supervisory Control in Discrete-Event Systems

Supervisory control theory of DES was first proposed by Ramadge and Wonham [12]. For a plant automaton $G = (Q, \Sigma, \delta, q_0)$, the event set Σ is partitioned into two disjoint subsets $\Sigma = \Sigma_c \cup \Sigma_{uc}$ where Σ_c is the set of *controllable events* and Σ_{uc} is the set of *uncontrollable events*.

In [12], the control objective, called the *(language) specification*, is defined by a regular language $K \subseteq \Sigma^*$. A supervisor S that dynamically disables events of the plant such that the closed-loop language of S over G is restricted within K. Here we use S/G to denote the closed-loop system composed by the plant G under the supervision of S, and we use $L(S/G)$ to denote the language of S/G. A supervisor S runs in parallel with the plant and, when a plant generates a sequence $s \in L(G)$, makes a control decision $\xi(s) \subseteq \Sigma_c$ that is to disable all controllable events not in $\xi(s)$. Note that a supervisor cannot disable any uncontrollable $\sigma \in \Sigma_{uc}$. Supervisor S can be represented by an automaton $S = (Q_S, \Sigma, \delta_S, q_{s,0})$. For a sequence $s \in L(S)$, the control decision $\xi(s)$ is given by: $\xi(s) = \Sigma_c \cap \Gamma(\delta_S(q_{s,0}, s))$, i.e., S disables all controllable events not defined at the current supervisor state $\delta_S(q_{s,0}, s)$. For details of the synthesis of S please refer to [16].

A language K is said to be *controllable* with respect to $L(G)$ if $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$ holds. If K is not controllable, a supervisor S enforcing K can be obtained by computing the *supremal controllable sublanguage* [16] of $L(G)$ with respect to K, i.e.:

$$K^{\uparrow C} = \bigcup \{H \subseteq K \mid H \text{ is controllable to } L(G)\}.$$

by iterative manipulations on regular languages.

A *state specification* defines a set of forbidden states $Q_F \subseteq Q$ that requires that the plant does not reach any state in Q_F. A supervisor S that enforces Q_F can be similarly obtained by first converting the state specification Q_F into its equivalent language specification $K = \{s \in L(G) \mid \delta(q_0, s) \notin Q_F\}$ followed by computing the supremal controllable sublanguage of K.[1]

III. Resilient Supervisors Against Indefinite Actuator Attacks

In this paper, a plant is modeled by an automaton $G = (Q, \Sigma, \delta, q_0)$ with $\Sigma = \Sigma_c \cup \Sigma_{uc}$ where Σ_c, Σ_{uc} are the sets of controllable and uncontrollable events, respectively. In G some states are considered to be unsafe and should be prevented from entering. When there is no intruder, a maximally permissive supervisor S can be obtained by computing the supremal controllable sublanguage $K^{\uparrow C}$ with respect to $L(G)$ using standard supervisory control paradigm [12]. Due to the existence of an intruder performing actuator attacks, however, a supervisor S that enforces $K^{\uparrow C}$ may not be resilient against such attacks. An actuator attack (illegally) allows disabled event to execute, which may result in a plant sequence that is outside of $L(S/G) = K^{\uparrow C}$ and eventually violates the specification K. Hence, a resilient supervisor which will be denoted by S_r must guarantee that even if some events are attacked, the supervisor is still able to stop the plant from reaching the unsafe states.

In the literature, it is assumed that some controllable events are invincible to attacks; other events are vulnerable which can

[1]Marking and nonblocking supervisory control can also be considered with no affect on the problem/solution developed below.

This article has been accepted for publication in IEEE Control Systems Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2022.3168926

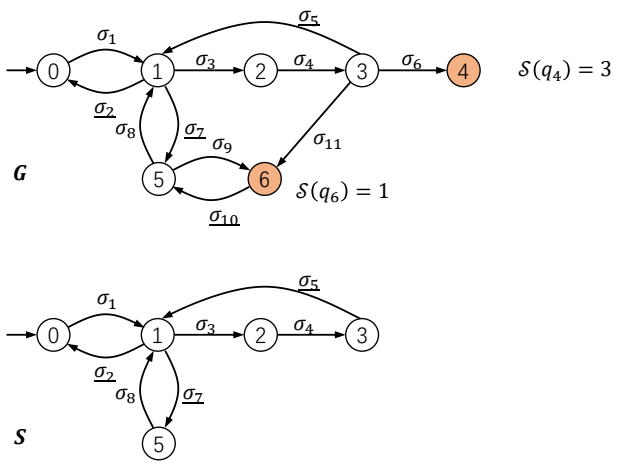

Fig. 1. Plant and supervisor in Example 1

be attacked infinitely often. However, as we have mentioned in Section I, this assumption may not be satisfied in practice. On one hand, which events are subject to malicious attacks may well be indefinite (unknown *a priori* to the designer). On the other hand, an intruder may not be able to perform actuator attacks infinitely often but only within a limited window before it is detected and wiped out from the system. Therefore, the aim of resilient control can be achieved by setting a barrier for the intruder such that the intruder has to attack a certain number of events in order to drive the plant to an unsafe state. If the barrier is sufficiently high, the potential attack will be practically prevented.

Definition 1: Given a plant $G = (Q, \Sigma, \delta, q_0)$, a *safety requirement* is a function $\mathcal{S} : Q \to \mathbb{N}$ that assigns to each state $q \in Q$ a *safety level* $\mathcal{S}(q)$.

In plain words, $\mathcal{S}(q)$ means that to reach state q an intruder must (successfully) attack at least $\mathcal{S}(q)$ controllable events. Particular attention is given to those unsafe states, the safety level of which can be viewed as a measure of importance for the plant to avoid entering them. Note that $\mathcal{S}(q) > 0$ if and only if state q is unsafe. We illustrate this point by the following example.

Example 1: Consider the plant G in Figure 1 which is an abstracted model of a zone-controlled auto-guided vehicle, i.e., an vehicle run following guide-routines to visit different locations. In G, the set of controllable events is $\Sigma_c = \{\sigma_1, \sigma_3, \sigma_4, \sigma_6, \sigma_8, \sigma_9, \sigma_{11}\}$ and the set of controllable ones is $\Sigma_{uc} = \{\sigma_2, \sigma_5, \sigma_7, \sigma_{10}\}$ (underlined). Assume that states q_4 and q_6 are unsafe states. The conventional supervisor S enforcing $K^{\uparrow C}$ is shown in Figure 1.

Suppose that we have a safety requirement $\mathcal{S}(q_4) = 3, \mathcal{S}(q_6) = 1$, and $\mathcal{S}(q_i) = 0$ for all $i \neq 4, 6$; that is, from the viewpoint of the system designer, it is acceptable that the intruder must successfully attack three actuators in order to make the system enter q_4 and one actuator to enter q_6. It is not difficult to see that supervisor S cannot achieve this resilient control aim. Indeed, supervisor S allows the plant to reach state q_3 by executing $\sigma_1\sigma_3\sigma_4$, where event σ_6 is disabled. If the intruder succeeded an actuator attack to enable σ_6 at this moment, the plant will reach state q_4 by just one actuator attack.

Remark 1: The value of $\mathcal{S}(q)$ for each state q is dependent on the crucialness of state q. For example, consider the following two unsafe states: (i) an AGV enters an empty zone without authorization, (ii) two AGVs run on the same track from opposite directions. The first unsafe state is undesirable but is no harm, since even if such situation happens, it is easy to instruct the AGV to quit the zone. On the other hand, the latter unsafe state is rather fatal since the two AGVs may crash. Hence, we may expect a resilient supervisor to ensure that an intruder must (successfully) attack at least one actuator before reaching unsafe state 1 while attack at least two actuators before reaching unsafe state 2.

Before we proceed, we note a key difference between the resilient control aim in this work and the conventional control specification. In the conventional supervisory control paradigm, the actuator attack is not considered and the unsafe states are strictly forbidden. Here, since the actuator attacks exist and are indefinite, it is in general not possible to achieve the conventional goal. Instead, we design a resilient supervisor that continues to function when attack occurs. Essentially, the resilient supervisor sets a sufficiently high barrier to practically prevent the intruder from making the system reach those unsafe states.

Notice that a conventional supervisor (see Example 1) cannot make control decisions when a sequence not belonging to $L(S)$ is executed: such a sequence may comes from the illegal re-enabling of disabled events. This indicates that a resilient supervisor should contain both the normal and the attacked behavior of the system to function after observing the attacks. Given a plant G whose set of controllable events is Σ_c, we define Σ_a to be a duplicate of Σ_c, i.e., for $\Sigma_c = \{\sigma_1, \ldots, \sigma_n\}$, $\Sigma_a = \{\sigma_{1,a}, \ldots, \sigma_{n,a}\}$. (Formally, Σ_c and Σ_a are disjoint and isomorphic.) The resilient supervisor is then defined as the following.

Definition 2: Given a plant $G = (Q, \Sigma, \delta, q_0)$ with $\Sigma = \Sigma_c \cup \Sigma_{uc}$, a *resilient supervisor* is an automaton $S_r = (Q_r, \Sigma \cup \Sigma_a, \delta_r, q_{r,0})$ where Σ_a is a duplicate of the controllable event set Σ_c. For a sequence $s \in L(S_r)$, the control decision $\xi_r(s)$ is given by: $\xi_r(s) = \Sigma_c \cap \Gamma(\delta_r(q_{r,0}, s))$, i.e., S disables all controllable events in Σ_c not defined by δ_r at the current supervisor state $\delta_r(q_{r,0}, s)$.

The feature of a resilient supervisor S_r is that it contains a type of transitions labeled by Σ_a called the "attacked transitions". These attacked transitions are not used for issuing control decisions but to let S_r correctly react to the actuator attacks. When the plant executes an event $\sigma_i \in \Sigma$, S_r execute the same event σ_i as in the conventional control paradigm. On the other hand, when the plant executes an event σ_i that is supposed to be disabled by S_r, the supervisor 'knows' that an attack has occurred. Thus S_r executes the corresponding attacked event $\sigma_{i,a}$ and continues to issue control actions accordingly.

Example 2: Consider the plant G (with $\Sigma_c = \{\sigma_2, \sigma_3\}$) and two supervisors S, S_r in Figure 2, where the plant state 2 is unsafe and has a safety level 2, i.e., $\mathcal{S}(q_2) = 2$. One can intuitively design a conventional supervisor S that disables

This article has been accepted for publication in IEEE Control Systems Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2022.3168926

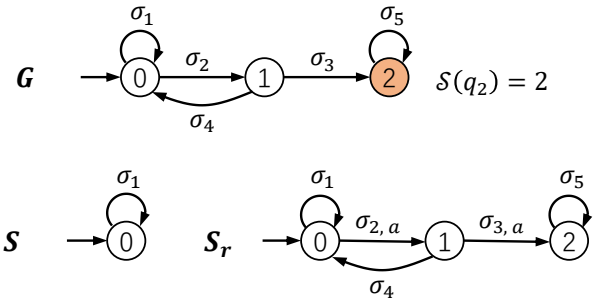

Fig. 2. Plant G, supervisor S, and resilient supervisor S_r in Example 2.

event σ_2 at the initial state (note that S is smaller than the conventional maximally permissive supervisor: the latter cannot enforce the specification). However, if event σ_2 is attacked/re-enabled by an intruder, S cannot respond to the execution of σ_2 and hence ceases to function.

On the other hand, the resilient supervisor S_r contains a transition labeled by $\sigma_{2,a}$ which represents the "attacked event σ_2". Hence, when there is no attack, S_r disables σ_2, which is the same control action as S does. When σ_2 is attacked, by observing the execution of σ_2, S_r moves to its state 1 by executing $\sigma_{2,a}$ where it disables event σ_3.

We propose some notions useful to formalize the problem. Given a resilient supervisor S_r, a sequence $s \in L(S_r)$, and an event $\sigma_i \in \Sigma$, we define $D(s, \sigma_i) = 1$ if $\delta(\delta(q_0, s), \sigma_i)$ is defined but $\delta_r(\delta_r(q_{r,0}, s), \sigma_i)$ is *not* defined (i.e., disabled by S_r), and $D(s, \sigma_i) = 0$ otherwise. Then, for a plant sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$, we define $\ell(s) = \sigma'_1 \cdots \sigma'_n$ as:

$$(\forall i = 1, \ldots, n)\ \sigma'_i = \begin{cases} \sigma_i, & D(\sigma_1 \cdots \sigma_{i-1}, \sigma_i) = 0 \\ \sigma_{i,a}, & D(\sigma_1 \cdots \sigma_{i-1}, \sigma_i) = 1 \end{cases} \quad (1)$$

(Here $\sigma_{-1} := \varepsilon$.) In plain words, when the plant executes a sequence s, the resilient supervisor S_r simultaneously executes $\ell(s)$ — a sequence obtained from s such that each attacked event σ_i is replaced by $\sigma_{i,a}$. For example, in Figure 2 when G executes $s = \sigma_1\sigma_2\sigma_4$, S_r executes $\ell(s) = \sigma_1\sigma_{2,a}\sigma_4$ since event σ_2 is disabled after executing σ_1. Now we are ready to formalize the problem studied in this paper as the following.

Problem 1: Given a plant $G = (Q, \Sigma, \delta, q_0)$ and a safety requirement $\mathcal{S} : Q \to \mathbb{N}$, construct a resilient supervisor S_r that enforces $\mathcal{S}$, i.e., for every q with $\mathcal{S}(q) > 0$ and every sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$ such that $\delta(q_0, s) = q$, there holds:

$$\sum_{i=0}^{n-1} D(\ell(\bar{s}_i), \sigma_{i+1}) \geq \mathcal{S}(q). \quad (2)$$

We note that the methods in the literature are incomparable with our setting. Since we allow all events to be attacked (while instead set an additional constraint on the number of attacks), the methods in [2], [6], [8], [11] return no solution.

IV. Synthesis of Resilient Supervisors Using Resiliency Automaton

A. The Intruder Automaton

Given a safety requirement $\mathcal{S} : Q \to \mathbb{N}$, let k be the maximum safety level in $\mathcal{S}$, i.e., $k = \max_{q \in Q} \mathcal{S}(q)$. Evidently, to achieve this safety requirement, we do not need to consider attacks on more than k controllable events. The relevant behavior of the attack intruder can be described by an automaton which we call the *Intruder Automaton* (IA) $A = (Q_A, \Sigma_A, \delta_A, q_{a,0})$ where:

- $Q_A = \{q_{a,0}, \ldots, q_{a,k-1}\}$ is a set of states;
- $\Sigma_A = \Sigma \cup \Sigma_a$ is the event set, where Σ is the plant event set in G and $\Sigma_a = \{\sigma_{1,a}, \ldots, \sigma_{n,a}\}$ is the duplicate of the controllable event set Σ_c of the plant;
- $q_{a,0}$ is the initial state;
- $\delta_A : Q_A \times \Sigma_A \to Q_A$ is the partial transition function:

$$\delta_A(q_{a,i}, \sigma) = \begin{cases} q_{a,i}, & \sigma \in \Sigma, i = 0, \ldots, k-1 \\ q_{a,i+1}, & \sigma \in \Sigma_a, i = 0, \ldots, k-2 \end{cases}$$

Intuitively speaking, the IA contains two types of events. Events in Σ are plant events not attacked by the intruder. On the other hand, each event $\sigma_{i,a}$ in Σ_a represents the controllable event $\sigma_i \in \Sigma_c$ attacked by the intruder.

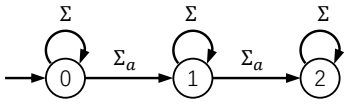

Fig. 3. The intruder automaton in Example 3.

B. Resiliency automaton

We now define an operator called the *A-synchronization*. The A-synchronization can be viewed as a special kind of parallel synchronization that yields a model of plant under attack.

Definition 3: Given a plant $G = (Q, \Sigma, \delta, q_0)$ and an IA $A = (Q_A, \Sigma_A, \delta_A, q_{a,0})$ of it, the *A-synchronization* of G and A, denoted as $H = G \odot A$, is an automaton $H = Ac(Q \times Q_A, \Sigma \cup \Sigma_a, \delta_H, (q_0, q_{a,0}))$ where the transition function is

$$\begin{cases} \delta_H((q, q_{a,i}), \sigma) = (\delta(q, \sigma), \delta_A(q_{a,i}, \sigma)) \\ \delta_H((q, q_{a,i}), \sigma_a) = (\delta(q, \sigma), \delta_A(q_{a,i}, \sigma_a)) \end{cases}$$

if σ is defined at state q in G and σ_a is the corresponding *attacked* event. Automaton H is called the *resiliency automaton* (RA) of G and A.

Intuitively speaking, in the resiliency automaton H (A-synchronized by G and A), the attacked behavior of the system is encoded. A state $(q, q_{a,i})$ in H means that the plant is at state q while i actuator attacks have occurred.

Example 3: Again consider the plant automaton G in Figure 1 with $\mathcal{S}(q_4) = 3$, $\mathcal{S}(q_6) = 1$, and $\mathcal{S}(q_i) = 0$ for other states. The intruder automaton A is depicted in Figure 3. The resiliency automaton $H = G \odot A$ is depicted in Figure 4. In H, transition $\delta_H((q_0, q_{a,0}), \sigma_1) = (q_1, q_{a,0})$ means: if the plant is at state q_0 and event σ_1 is executed normally (i.e., it is not attacked), then by executing σ_1 the plant moves to state q_1 while the attack counter remains 0. On the other hand, transition $\delta_H((q_0, q_{a,0}), \sigma_{a,1}) = (q_1, q_{a,1})$ means that if σ_1 is disabled but attacked/re-enabled by the intruder, by executing $\sigma_{a,1}$ the plant moves to state q_1 while the attack counter increases to 1. ♢

This article has been accepted for publication in IEEE Control Systems Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2022.3168926

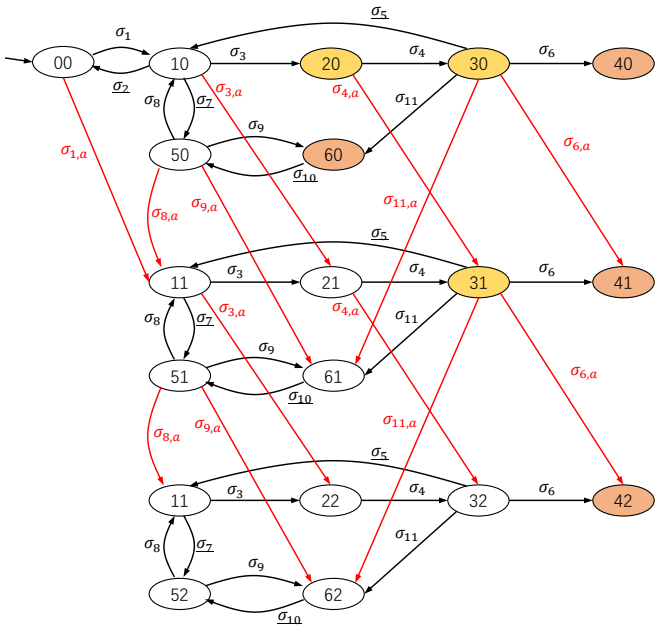

Fig. 4. The resiliency automaton $\boldsymbol{H}$ in Example 3. Notation: state $(q_i, q_{a,j})$ is denoted as "ij".

C. Resilient Supervisor Design Using Supervisor Control

In this subsection, we show the main result of this work; that is, a resilient supervisor of G can be derived by solving a conventional supervisory control problem in the RA. This problem in the RA is formulated as follows.

Problem 2: Given a plant $G = (Q, \Sigma, \delta, q_0)$, a safety requirement $\mathcal{S} : Q \to \mathbb{N}$, and the corresponding RA $H = (Q \times Q_A, \Sigma \cup \Sigma_a, \delta_H, (q_0, q_{a,0}))$, construct a (conventional) supervisor S for H to enforce the state specification:

$$Q_F = \{(q, q_{a,i}) \in Q \times Q_A \mid i < \mathcal{S}(q)\}$$

(set of forbidden states) with respect to controllable transitions Σ_c and uncontrollable transitions $\Sigma_{uc} \cup \Sigma_a$.

Problem 2 is a standard supervisory control problem which can be solved using the Ramadge-Wonham paradigm. This will illustrated in Example 4 shortly. Then, by the following theorem we show that the resilient supervisor design problem, i.e., Problem 1, can be reduced to Problem 2.

Theorem 1: A resilient supervisor $S_r = (Q_r, \Sigma \cup \Sigma_a, \delta_r, q_{r,0})$ for plant G that solves Problem 1 if and only if it is a conventional supervisor[2] for the RA H that solves Problem 2.

Proof: ($\Rightarrow$) Let S_r be a valid resilient supervisor that enforces the safety requirement $\mathcal{S}$ for the plant G. We prove that S_r is also a solution to Problem 2. By contraposition, suppose that S_r is not a solution of Problem 2, i.e., S_r permits a sequence $s \in L(H)$ to reach a forbidden state $(q, q_{a,i}) \in Q_F$ with $i < \mathcal{S}(q)$. According to the construction of the RA H, the subscript i in state $(q, q_{a,i})$ indicates that the number of attacked events in s is i that is less than $\mathcal{S}(q)$, i.e., $\sum_{i=0}^{n-1} D(\bar{s}_i, \sigma_{i+1}) < \mathcal{S}(q)$, which implies that S_r is not a valid resilient supervisor for G. This is a contradiction.

($\Leftarrow$) Let S_r be a solution of Problem 2. Suppose on the contrary that S_r is not a resilient supervisor of Problem 1. Then there exists a sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$ such that $\sum_{i=0}^{n-1} D(\bar{s}_i, \sigma_{i+1}) < \mathcal{S}(q), q = \delta(q_0, s)$. It means that in H by executing s a state $(q, q_{a,i})$ with $i < \mathcal{S}(q)$ is reached, i.e., supervisor S_r permits s in H. This indicates that S_r violates the state specification in Problem 2 and is thus not a valid solution. This is a contradiction. ■

Since G is fully observable and every attack causes an abnormal behavior, the RA H is also fully observable. Hence, by [12] there exists a unique maximally permissive supervisor $\hat{S}_r$ for Problem 2. Next, we show that such a supervisor $\hat{S}_r$ of Problem 2 is also the maximally permissive resilient supervisor of Problem 1.

Definition 4: A resilient supervisor S_r is *maximally permissive* if for any other resilient supervisor $S_r' \neq S_r$ and any sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$, $\xi_r(s) \supseteq \xi_r'(s)$ holds, where ξ_r, ξ_r' are the control decisions made by S_r and S_r'.

Theorem 2: A maximally permissive supervisor $\hat{S}_r$ for Problem 2 is a maximally permissive resilient supervisor for Problem 1.

Proof: By contraposition, suppose that a maximally permissive supervisor S_r for Problem 2 is not a maximally permissive resilient supervisor for Problem 1. By Definition 4 there exists another resilient supervisor S_r for Problem 1 and a sequence $s = \sigma_1 \cdots \sigma_n$ such that $\xi_r(\bar{s}_k) = \xi_r'(\bar{s}_k)$ holds for all $k \in \{1, \ldots, n-2\}$, and $\xi_r(\bar{s}_{n-1}) = \xi_r'(\bar{s}_{n-1}) \setminus \{\sigma_n\}$ holds. In other words, the control decisions of S_r and S_r' are the same for the first $n-1$ events in s, while S_r disables the last event σ_n in s while S_r' does not. By Theorem 1, both S_r, S_r' are solutions of Problem 2, and S_r disallows s in H while S_r' permits s. This contradicts the assumption that S_r is a maximally permissive supervisor for Problem 2. ■

Example 4: Again consider the resiliency automaton H in Figure 4. We construct Problem 2 as follows. For specification $\mathcal{S}(q_4) = 3$, $\mathcal{S}(q_6) = 1$, and $\mathcal{S}(q_i) = 0$ for other states, the forbidden states are (in orange) $(q_4, q_{a,0})$, $(q_4, q_{a,1})$, $(q_4, q_{a,2})$, $(q_6, q_{a,0})$. The uncontrollable transitions are those Σ_{uc} (underlined) and those in Σ_a (in red). Then, such a problem can be solved by the Ramadge-Wonham paradigm so that the maximally permissive supervisor S_r is shown in Figure 5: according to Theorems 1 and 2 it is the maximally permissive resilient supervisor S_r enforcing $\mathcal{S}$ for G.

Let us see how S_r works. Initially, S_r permits σ_1. By executing σ_1 the plant reaches state q_1 while S_r moves to state $(q_1, q_{a,0})$ where event σ_3 is disabled. Suppose that now an actuator attack occurs which re-enables σ_3. By noticing the unexpected execution of σ_3, i.e., $\sigma_{a,3}$, S_r moves to state $(q_2, q_{a,1})$ and tries to disable σ_4 to let G stay at q_2. When the second attack re-enables and executes σ_4 , S_r moves to state $(q_3, q_{a,2})$ where it tries to disable σ_6 while permits σ_{11} and σ_5 (the latter is uncontrollable) to provide the plant a way to go to state q_1 or state q_6. In any case, state q_4 will not be reached if less than three actuator attacks are preformed.

Note that after the second attack at supervisor state $(q_3, q_{a,2})$ (plant state q_3) event σ_{11} is enabled, which indicates that the

[2]When S_r is viewed as a conventional supervisor, all of its events are treated as normal events, i.e. not attacked.

This article has been accepted for publication in IEEE Control Systems Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2022.3168926

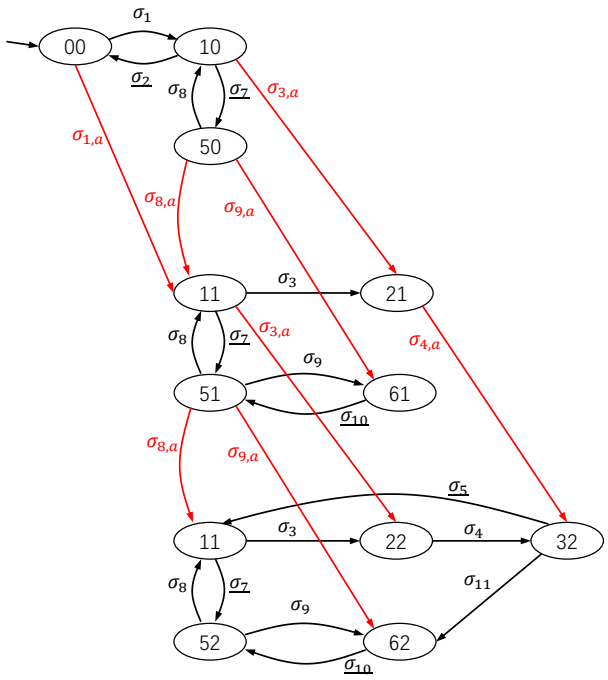

Fig. 5. Resilient supervisor S_r in Example 4. Notation: state $(q_i, q_{a,j})$ is denoted as "ij".

plant may reach unsafe state q_6 by executing σ_{11}. However, this does not violate the safety specification $\mathcal{S}$. Since $\mathcal{S}(q_6) = 1$, from the viewpoint of the designer of the plant it is acceptable that the plant reaches state 6 after one actuator attack (in this case two attacks have happened).

At the end of this section we have the following remarks. First, by Theorem 1, a valid resilient supervisor exists if and only if the corresponding supervisory control problem in the RA has a solution, i.e., the maximally permissive $\hat{S}_r$ for Problem 2 is not empty. Hence, the existence of a resilient supervisor for G and safety requirement $\mathcal{S}$ can be verified by checking the emptiness of the corresponding supremal controllable sublanguage of the resiliency automaton H with respect to the state specification. According to its definition, resiliency automaton H has at most $k \cdot |Q|$ states and at most twice of the transition numbers of G. Since checking the emptiness of a supremal controllable sublanguage is polynomial ([16] Chapter 3), we conclude that the complexity of designing a resilient supervisor is also polynomial.

Second, although so far we have focused on fully observable systems, we point out that Theorems 1 and 2 also hold for partially observable plants. When plant G is partially observable, a maximally permissive resilient supervisor can still be obtained by solving Problem 2 in the corresponding resiliency automaton H. In such a case, however, there may not exist a maximally permissive supervisor but several incomparable locally maximal solutions. The latter can be obtained by using synthesis techniques in [14], [18].

V. Conclusion

In this paper we have studied the resilient controller design problem in DES against indefinite actuator attacks. We have proposed a new structure called the resiliency automata, based on which we have developed a polynomial method to design a resilient supervisor such that the plant under control does not reach any unsafe state by a number of actuator attacks no greater than the required safety level of the state. In future work, we will investigate this problem in a decentralized/hierarchical setting.

References

[1] M. Agarwal. Rogue twin attack detection: A discrete event system paradigm approach. In *Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 1813–1818, Oct 2019.

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

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

[4] R. Fritz and P. Zhang. Modeling and detection of cyber attacks on discrete event systems. In *Proceedings of the 14th IFAC Workshop on Discrete Event Systems*, pages 285–290, Sorrento, Italy, 2018.

[5] C. N. Hadjicostis. *Estimation and Inference in Discrete Event Systems — A Model-Based Approach with Finite Automata*. Springer-Verlag US, 2020.

[6] R. Su L. Lin, Y. Zhu. Synthesis of covert actuator attackers for free. *Discrete Event Dynamic Systems:Theory and Applications*, 2020.

[7] L. Lin and R. Su. Synthesis of covert actuator and sensor attackers. *Automatica*, 130:Article 109714, 2021.

[8] L. Lin, S. Thuijsman, Y. Zhu, S. Ware, R. Su, and M. Reniers. Synthesis of supremal successful normal actuator attackers on normal supervisors. In *2019 American Control Conference*, pages 5614–5619, 2019.

[9] J. P. Hespanha M. Wakaiki, P. Tabuada. Supervisory control of discrete-event systems under attacks. *Dynamic Games and Applications*, 9:965–983, 2019.

[10] Z. Ma, X. Yin, and Z. Li. Verification and enforcement of strong infinite- and k-step opacity using state recognizers. *Automatica*, 133:Article 109838, 2021.

[11] R. Meira-Goes, E. Kang, R. Kwong, and S. Lafortune. Synthesis of sensor deception attacks at the supervisory layer of cyber-physical systems. *Automatica*, 121:Article 109172, 2020.

[12] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of IEEE*, 77(1):81–98, 1989.

[13] A. Saboori and C. N. Hadjicostis. Verification of k-step opacity and analysis of its complexity. *IEEE Transactions on Automation Science and Engineering*, 8(3):549–559, 2011.

[14] S. Takai. Synthesis of maximally permissive supervisors for nondeterministic discrete event systems with nondeterministic specifications. *IEEE Transactions on Automatic Control*, page in press, 2020.

[15] Y. Wang and M. Pajic. Supervisory control of discrete event systems in the presence of sensor and actuator attacks. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 5350–5355, 2019.

[16] W. M. Wonham and Kai Cai. *Supervisory Control of Discrete-Event Systems*. Springer, 2019.

[17] Y.-C. Wu and S. Lafortune. Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, 23(3):307–339, 2013.

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

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

[20] Q. Zhang, C. Seatzu, Z. Li, and A. Giua. Joint state estimation under attack of discrete event systems. *IEEE Access*, 9:168068–168079, 2021.