Attacks on Distributed Sequential Control in Manufacturing Automation

Zivana Jakovljevic, *Member, IEEE*, Vuk Lesi, and Miroslav Pajic, *Senior Member, IEEE*

***Abstract*—Industrial Internet of Things (IIoT) represents a backbone of modern reconfigurable manufacturing systems (RMS), which enable manufacturing of a high product variety through rapid and easy reconfiguration of manufacturing equipment. In IIoT-enabled RMS, modular equipment is built from smart devices, each performing its own tasks, whereas the global functioning is achieved through their networking and intensive communication. Although device communication contributes to the system reconfigurability, it also opens up new security challenges due to potential vulnerability of communication links. In this article, we present security analysis for a major part of RMS in which manufacturing equipment is sequentially controlled and can be modeled as discrete event systems (DES). Control distribution within DES implies communication of certain events between smart modules. Specifically, in this work, we focus on attacks on communication of these events. In particular, we develop a method for modeling such attacks, including event insertion and removal attacks, in distributed sequential control; the method is based on the *supervisory control theory* framework. We show how the modeled attacks can be detected and provide a method for identification of communication links that require protection to avoid catastrophic damage of the system. Finally, we illustrate and experimentally validate applicability of our methodology on a real-world industrial case study with reconfigurable manufacturing equipment.**

I. Introduction

INDUSTRIAL implementation of Internet of Things (IoT) and cyber–physical systems (CPS) significantly changes the way we manufacture, leading to the evolution of manufacturing systems to a new level known as Industry 4.0 [1]. Industry 4.0 factory is a smart factory able to meet the requirements of each individual customer through implementation of reconfigurable manufacturing systems (RMS) [2]. RMS are based on modular

Manuscript received February 19, 2020; accepted March 30, 2020. Date of publication April 15, 2020; date of current version November 18, 2020. This work was supported in part by the ONR under Grant N00014-17-1-2504 and Grant N00014-20-1-2745, in part by AFOSR under Award FA9550-19-1-0169, in part by the NSF under Award CNS-1652544, and in part by the Serbian Ministry of Education, Science and Technology under Grant TR35004 and Grant TR35020. Paper no. TII-20-0860. *(Corresponding author: Zivana Jakovljevic.)*

Zivana Jakovljevic is with the Faculty of Mechanical Engineering, University of Belgrade, Belgrade 11000, Serbia (e-mail: zjakovljevic@mas.bg.ac.rs).

Vuk Lesi and Miroslav Pajic are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: vuk.lesi@duke.edu; miroslav.pajic@duke.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.2987629

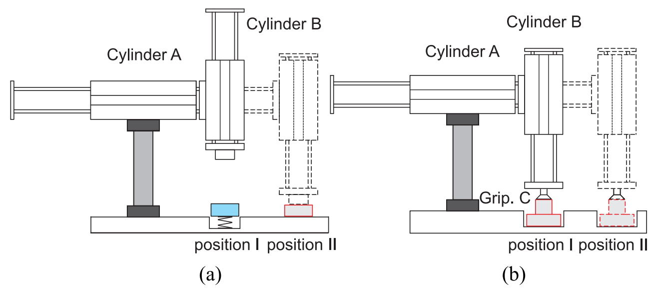

Fig. 1. Examples of reconfigurable manufacturing equipment. (a) Running example: Configuration of the system for parts marking. (b) Case study: Configuration of the system for parts manipulation.

equipment that is physically and functionally reconfigurable and can be rapidly and easily adapted to manufacturing of different products [3], [4]; Fig. 1 presents an example of a reconfigurable pneumatic device. To facilitate reconfigurability, the modularity should be achieved not only in terms of mechanical elements, but also in equipment/tool control, where each mechanical module is augmented by its own local controller (LC) with communication and computation capability, representing a smart IoT device.

Control system modularity leads to a shift from the classical IEC 62264 hierarchical industrial automation pyramid to distributed control systems [1], where control is realized through peer-to-peer communication of networked devices that create industrial IoT (IIoT) [5]. In distributed control of manufacturing systems, each control task is realized through coordinated operations of a number of smart devices that comprise the considered reconfigurable equipment, with the corresponding LCs communicating relevant information to each other in order to achieve the desired system behavior. On the other hand (usually wireless), communication between LCs introduces new security challenges [6] since communication link may be prone to attacks by adversaries.

In IIoT systems, end-to-end (including communication) security guarantees are of crucial importance [7]. There are different ways to protect communication between devices, such as the use of cryptographic mechanisms to provide continuous or intermittent authentication or adding watermarking/random noise signals (e.g., [8]–[10]). Yet, all such methods introduce additional computation/communication overhead, increase communication latency [11], and should be applied only when necessary in resource constrained IIoT-enabled RMS.

Different types of cyberattacks have been reported (e.g., in [12]), including replay attacks where attacker records

1551-3203 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

sensor/actuator signals in one period of time and replays them in another, or covert attacks where adversary secretly takes over control from the supervisor, with the goal to remain undetected. For all attacks, it is common that they are not random (opposite to failures) and that adversaries are deceptive and insidious in their goals—e.g., intention to remain stealthy and to achieve negative effect on the system performance. Usually the attackers have some *a priori* knowledge about the system obtained through different cyber–physical intelligence attacks [13], such as eavesdropping.

While the attacks in continuous-time control systems [12], [13] have gained significant attention, attacks in discrete event systems (DES) were only recently explored [14]–[24]. Supervisory control theory (SCT) models DES as generators of formal languages whose behavior can be captured by finite state machines (FSM) [25]. Since SCT and FSM were successfully employed for fault detection in DES, their application in studying DES attacks, as done in this article, represents a natural extension. An approach for modeling and detection of actuator enablement/disablement and sensor removal/insertion attacks in remotely supervised plants is presented in [14]. System under attacks is modeled using FSM and SCT frameworks, whereas attacks detection and prevention of system from reaching unsafe state is based on DES fault diagnosis. Similar approach for man-in-the-middle sensor attacks is presented in [15], whereas the defense strategy for attacks from the work in [14] and [15] is given in [26]. Furthermore, Lima *et al.* [22] provides the mechanisms for implementation of security modules for the attacks from the work in [15].

Intelligent adversary with *a priori* knowledge about supervisor‘s performance that arbitrary alters sensors’ readings is modeled in [16], as well as a supervisor robust to these attacks. [17] models event insertion/removal attacks as SCT-based projections that map observed into corrupted events strings through events replacing or inserting, whereas [18] studies replay and covert attacks in DES and proposes the detection method based on permutation of controller inputs and outputs on the plant and supervisor side. In addition, Fritz *et al.* [23] consider the attacks that completely take over the control over plant for a certain time period. Furthermore, Zhang *et al.* [19] and Ges *et al.* [20] propose methods for design of stealthy attacks in such systems. Recent review of the state of the art in application of SCT and FSM in DES attacks modeling and detection is given in [24].

Existing works in modeling and analysis of attacks on DES consider attacks on sensor and actuator signals in the case of a remote plant and a supervisor that carries out *centralized* control (e.g., [17], [21], and [24]). On the other hand, the distribution of control tasks to smart devices within RMS and intensive communication between them bring about new security challenges. For example, each cylinder from Fig. 1(a) is a smart cylinder (with integrated limit switches and control valve) that is augmented by its own LC; the control of the system for parts marking is distributed over two LCs that intensively communicate, enabling control of the desired system behavior. In distributed sequential control for RMS, control can be captured as a DES [25]. In such systems, every IIoT-enabled LC is closely connected to the corresponding plant module, whereas signals (events) that are communicated between remote LCs (i.e., smart devices) may be vulnerable to attack.

Consequently, in this article, we focus on security analysis of *distributed* control systems for industrial automation, specifically addressing network-based attacks on event communication. To the best of our knowledge, these kinds of attacks have not been considered in the past. Attacks on communicated events in such systems could lead to an undesirable sequence of system actions, and the system should be prevented from generating unsafe sequence of events that can lead to catastrophic damage. We present an SCT-based modeling approach to capture common attacks—event insertion and removal in *distributed* sequential control. Furthermore, we introduce a method for attack detection and identification, focusing on safety-critical attacks that could violate safety requirements of system operation. To minimize computation and communication cost, we show how to determine a set of events whose communication should be protected to ensure safe system operation while minimizing security-related overhead.

Since our focus is on network-based attacks on sequential controllers in industrial automation systems, we are mainly considering impact on the automation due to false-data injection attacks as well as denial-of-service attacks, which prevent some of the messages from being delivered to the controllers.[1] Such attacks have been previously investigated in the other CPS domains where continuous control is applied, as in [27]–[29], where, e.g., attacks on power-grid infrastructure as well as on continuous control via SCADA systems were considered. On the other hand, we do not consider the origin of the attacks—e.g., the type of software/hardware vulnerability exploited by the attacker to launch the attack. The security-aware framework for industrial automation, which we introduce in this article, enables system designers to provide a formal proof about the attack-detectability and performance for the wide class of attacks, by employing a wide-range of tools for analysis of SCTs, such as [30].

The rest of this article is organized as follows. Section II briefly presents a method that is used for distribution of sequential controllers for RMS into LCs, whereas Section III maps such LCs into the SCT formalism. In Section IV, we present a method for attack modeling, which allows for the identification of events whose communication should be protected, which Section V further elaborates. In Section VI, the application of our security-aware methodology is presented on a real-world industrial case study. Finally, Section VII concludes this article.

II. Distributing Sequential Control Tasks to Smart Devices

Before considering security challenges in distributed sequential control, which are the topic of this article, we briefly outline the method from [31] that we use for distribution of control tasks to the LCs. We utilize this method since it is strongly related to the IEC 60848 and IEC 61131-3 standards that are commonly employed in practice for control specification. Furthermore, this

[1]On the other hand, since DES do not consider timing information, there is no need to address attacks that result in information only being delayed.

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

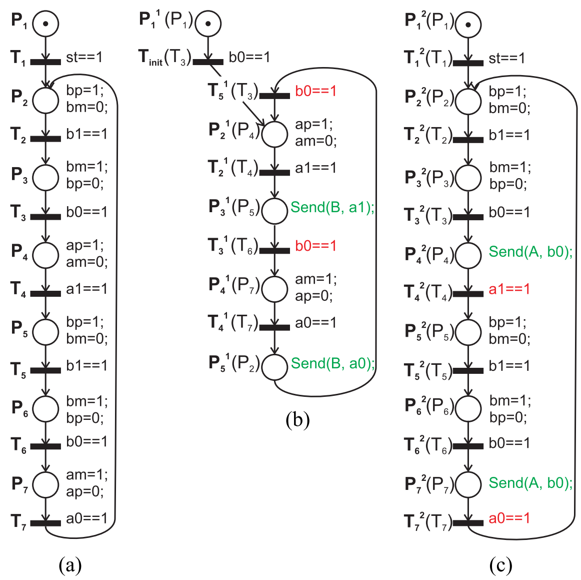

Fig. 2. Running example: (a) global **CIPN**, (b) **CIPN**$_1$ representing the behavior of LC$_1$, and (c) **CIPN**$_2$ representing the behavior of LC$_2$ (notation of places and transitions from **CIPN** are given in parentheses); $x == 1$ represents input reading allocated to the transition, whereas $x = 0/1$ denotes output assignment allocated to the place; Send commands are marked green, and receptive transition conditions red.

is a top-down approach, starting from a description of the system functionality as a whole and then distributing control tasks to LCs; thus, the representation of the LCs' functionalities and their relation to the overall control system is transparent and easily understandable. However, the results of this article (which considers attacks in distributed DES control) are not limited to the utilized method for distribution of control tasks and they can be applied to any distributed DES control regardless the way LCs are generated (using another approach, such as e.g., [32], or manually).

The method in [31] is based on control interpreted Petri nets (CIPNs) [33] that are captured as bipartite graphs with vertices referred to as places (denoted by P and graphically presented by circles) and transitions (denoted by T and graphically presented by bars), as illustrated in Fig. 2. The state of a CIPN is represented by a marking, which assigns one token to some of the places and which is dynamically changed by transitions firing. In CIPNs, transitions firings are synchronized with sensing events, whereas actuator outputs (commands) are issued from marked places.

The sequential control distribution starts from a CIPN-based high-level description of the desired system behavior when all sensors and actuators are connected to a centralized controller (referred to as *global* CIPN). Once a global CIPN is defined, and input and output signals are mapped into LCs with physical access to corresponding sensors and actuators, the method automatically generates local CIPN$_i$s, $i = 1, \ldots, N$ describing LCs executed on IIoT-enabled smart devices that communicate between each other to achieve coordination—e.g., `Send` commands in Fig. 2. We describe this in more detail using our running example, introduced ahead.

TABLE I
RUNNING EXAMPLE: SIGNALS MAPPING TO LCS

Cyl.	LC	Home sensor	End sensor	Cyl. adv. signal	Cyl. retr. signal	Other signals
A	LC$_1$	$a0$	$a1$	ap	am	-
B	LC$_2$	$b0$	$b1$	bp	bm	st

Example 1: We consider a system for parts marking shown in Fig. 1(a),[2] which consists of two double-acting cylinders (A and B) controlled by bistable dual control valves 5/2 (2 positions, 5 ports); the valves are activated/deactivated by signals introduced in Table I . Cylinders are also equipped with proximity sensors for detecting limit positions. System operation starts when the start switch (st in Table I) is pressed. The system's work cycle is described by the following sequence:

$$B+ \;\; B- \;\; A+ \;\; B+ \;\; B- \;\; A- \qquad (1)$$

where $X+$ denotes advancement and $X-$ retracting of cylinder X ($X \in \{A, B\}$). Cylinders represent smart devices with integrated LCs where the assignment of dual control valve activating signals and sensor signals to LCs is given in Table I.

From the behavior of system described in (1), we obtain a global **CIPN** shown in Fig. 2(a) that captures the functional specification for sequential control of the whole system. Using the procedure given in [31], from the global **CIPN**, we obtain each **CIPN**$_i$ describing local control behavior for LC$_i$ [see Fig. 2(b) and (c)], while ensuring the desired overall system behavior (as with the centralized controller). To achieve this, the LCs coordinate by communicating certain events. For example, LC$_2$ [see Fig. 2(c)], while at place P_4^2 (P_4), sends information about rising edge at $b0$ to LC$_1$ [see Fig. 2(b)] which receives this information at $T_{init}^1(T_3)$ or at $T_5^1(T_3)$, depending on the **CIPN**$_1$ marking and marks $P_2^1(P_4)$. In this way, the sequence T_3P_4 captured in the global **CIPN** [see Fig. 2(a)] is achieved in the distributed setup. ■

III. MODELING DISTRIBUTED SEQUENTIAL CONTROL

CIPNs are commonly used to model DES since they provide easily understandable graphical representation, especially in case of parallel processes. On the other hand, DES can also be represented as finite state automata (FSA). Since FSA provide convenient formalisms for modeling attacks on DES [34], in this work, we transform each **CIPN**$_i$ to FSA, utilizing procedures given in [35] and [36], within the SCT framework [25].

In SCT, all possible behaviors of a to-be-controlled-physical modules, which we will refer to as *plants* (e.g., cylinders in Fig. 1) can be represented as an FSA denoted by $G^i = (Q^i, E^i, f^i, q_0^i)$, where Q^i is the finite set of states, E^i is the finite set of events, $f^i : Q^i \times E^{i*} \rightarrow Q^i$ is the transition function (here, $*$ denotes Kleene star), and q_0^i denotes the initial state of G^i. Such plant can be regarded as a generator of a language $L^i(G^i)$ that contains strings w^i such that $L^i(G^i) := \{w^i \in E^{i*} : f^i(q_0^i, w^i)!\}$, where ! denotes that the $f^i(q^i, w^i)$ is defined.

[2]This system is similar to one of the systems used for illustration of control tasks distribution in [31].

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

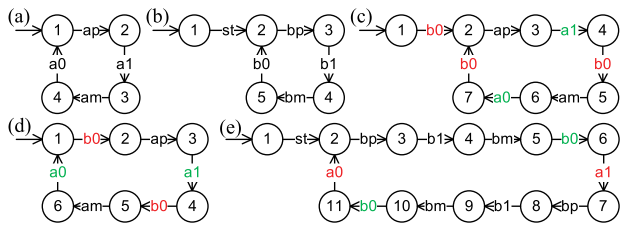

Fig. 3. Running example: (a) Automaton G^1 modeling behavior of cylinder A, (b) automaton G^2 representing cylinder B, (c) automaton $S^{1'}$ obtained from controller CIPN$_1$ [see Fig. 2(b)], (d) automaton S^1 equivalent to $S^{1'}$ representing LC$_1$, and (e) automaton S^2 representing LC$_2$ obtained from CIPN$_2$ [see Fig. 2(c)]. Events that supervisors send are marked green and the events that they receive are marked red.

Behavior of N plants within the system can be captured as the FSA G obtained by parallel composition of $G^i, i = 1, ..., N$, denoted by $G = ||_i G^i$.

For each plant, events in E^i can be partitioned as $E^i = E^i_o \cup E^i_{\text{uo}}$, where E^i_o and E^i_{uo} are the sets of observable and unobservable events, respectively. Similarly, the set E^i can be partitioned into the sets of controllable (E^i_c) and uncontrollable events (E^i_{uc}) such that $E^i = E^i_c \cup E^i_{\text{uc}}$. Since each physical plant modeled as G^i is locally controlled by an LC specified by $\mathbf{CIPN}_i$, sensor signals assigned to $\mathbf{CIPN}_i$ transitions belong to E^i_{uc}, while actuator signals assigned to the places are in E^i_c.

With distributed sequential control, LC$_i$ provides controlled behavior of the plant G^i through a feedback control loop by imposing supervisor S^i that restricts the language $L^i(G^i)$ by disabling certain events. Supervisor is only aware of observable events E^i_o obtained from the set E^i by the natural projection $P^i_o : E^{i*} \to E^{i*}_o$ where 1) $P^i_o(\epsilon) = \epsilon$, with ϵ denoting the empty string; and 2) $P^i_o(w^i t^i) = P^i_o(w^i)t^i$ if $t^i \in E^i_o$, and $P^i_o(w^i t^i) = P^i_o(w^i)$ if $t^i \notin E^i_o$. Such supervisor can be realized using automaton $S^i = (Q^i_s, E^i_s, f^i_s, q^i_{0s})$. Here, in addition to observable events from E^i, S^i contains events that are received (communicated) from other supervisors $S^j, j = 1, ..., i-1, i+1, ..., N$; we denote these events as $c^i_{j,k}$, where k denotes different events if more than one event is communicated from supervisor S^j to S^i. Thus, $E^i_s = E^i_o \cup \{\cup_j \cup_k c^i_{j,k}\}$. S^j transmits $c^i_{j,k}$ to S^i on the transition from state q^j_c for which $f^j(q^j_c, c^i_{j,k})!$ to the state $f^j(q^j_c, c^i_{j,k})$.

Finally, the coordinated operation of all supervisors S^i (i.e., all controllers) in the system is captured by $S = ||_i S^i$, while the controlled loop behavior of the system as a whole can be represented as $S \times G$, where $\times$ denotes the product operator.

Running example continued: All possible failure free behaviors of cylinders A and B are captured by automata G^1 and G^2 [see Fig. 3(a) and (b)], respectively. Here, $E^1 = E^1_o = \{ap, a1, am, a0\}$, with $E^1_c = \{ap, am\}$, and $E^2 = E^2_o = \{bp, b1, bm, b0, st\}$, with $E^2_c = \{bp, bm\}$. LC$_1$ and LC$_2$ implement supervisor controllers $S^{1'}$ and S^2, respectively, as shown in Fig. 3(c) and (e); these supervisors are obtained from $\mathbf{CIPN}_i$ in Fig. 2(b) and (c). To simplify the presentation, automaton $S^{1'}$ is replaced by equivalent automaton S^1 [see Fig. 3(d)][3]. To capture event communication between controllers LC$_1$ and LC$_2$, S^1 and S^2 have the following events sets: $E^1_s = \{ap, a1, am, a0, b0\}$ where $b0 = c^1_{2,1}$, and $E^2_s = \{bp, b1, bm, b0, st, a1, a0\}$ where $a0 = c^2_{1,1}$, and $a1 = c^2_{1,2}$. Communicated events are marked green in transmitting and red in receiving supervisor in Fig. 3(c)–(e); these events model `Send` commands from Fig. 2. The conjoint operation of S^1 and S^2—i.e., $S = S^1||S^2$—is graphically presented in Fig. 4. ■

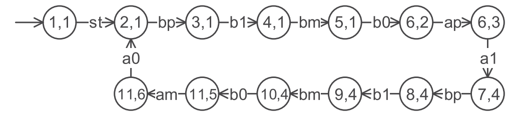

Fig. 4. Running example: Supervisor $S = S^1||S^2$; in state notation x, y: x, y refer to states from S^2 and S^1, respectively. In parallel composition, a transition on shared event can occur only if both automata are in a state where such transitions are enabled—e.g., transition on $b0$ (shared for S^1 and S^2) from (1,1) cannot occur as S^2 has no transition from (1) on $b0$.

IV. Modeling Impacts of Attacks in Distributed Sequential Control

In this article, we assume that the attacker may compromise events communicated between LCs. Using the LC representation from Section III, the compromised events for supervisor S^i are all the events that S^i receives from and transmits to other LCs, captured in sets $E^i_{r_x}$ and $E^i_{t_x}$:

$$E^i_{r_x} = \bigcup_j \bigcup_k c^i_{j,k} \subseteq E^i_s, \quad E^i_{t_x} = \bigcup_i \bigcup_k c^j_{i,k} \subseteq E^i_s. \qquad (2)$$

Since sequential control does not capture timing-related information and, thus, communication delays do not impact correctness of the system operation, in such systems, we have to consider two possible types of attacks: 1) *event insertion*, where a controller S^i receives an event $c^i_{j,k}$ before S^j sends it (i.e., without S^j sending it), and 2) *event removal*, where an event sent to a controller S^i from a controller S^j is not received. These attacks capture standard denial-of-service and false-data injection attacks [12], whereas attacks such as man-in-the-middle, which swap one event for another, can be obtained with a combination of these two attacks.

In this section, we focus on capturing impacts of such attacks on system operation. We assume that the attacker's goal is to affect the performance of the system without being immediately revealed; note that there is a number of attacks that can be easily detected, such as inserting events like $b0$ when automaton S^1 is in, e.g., state 3 [see Fig. 3(d)]. In addition, we assume that the attacker knows the current states of the plants and supervisors, and can use this information to plan his attacks. Finally, the attacker is not able to compromise protected communication links as integrity of these links is ensured with the use of

[3] All automata operations throughout the article are carried out in DESUMA software [30], where the equivalence of automata $S^{1'}$ and S^1 is checked.

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

standard cryptographic mechanisms for which the attacker does not possess the shared secret keys.

A. Insertion Attack

Let us consider the insertion attack that inserts event $c^i_{j,k} \in E^i_{r_x}$; to simplify our notation, we use s^i_r to denote the "regular" event ($c^i_{j,k}$) and s^i_a the event inserted by the attacker. To avoid being immediately revealed, the attacker has to insert event s^i_a only while the supervisor S^i is in a state q^i_r for which $f^i_s(q^i_r, s^i_r)!$. To achieve this, the attacker employs his knowledge of the current state of S^i.

Therefore, we will model the attacks that cannot be immediately revealed and that can affect the system behavior. Here, for every event that can be inserted, we need to capture effects of such attack on the supervisor that receives the event, and, as we describe ahead, modify the corresponding plant model to ensure that adding a new event does not prevent the plant model from evolving (as the plant-generated events are not directly affected by the inserted event). With attack event s^i_a that inserts s^i_r at state $q^i_r \in Q^i_s$ for which $f^i_s(q^i_r, s^i_r)!$, S^i transitions to the state $f^i_s(q^i_r, s^i_r)$ since S^i considers that real s^i_r is received. Thus, the LC$_i$ under attack can be modeled as automaton $S^i_a = (Q^i_s, E^i_{sa}, f^i_{sa}, q^i_{0s})$ with $E^i_{sa} = E^i_s \cup \{s^i_a\}$ and

$$f^i_{sa}(q^i, s^i) = \begin{cases} f^i_s(q^i, s^i), & \text{if } s^i \in E^i_s \text{ and } f^i_s(q^i, s^i)! \\ f^i_s(q^i, s^i_a), & \text{if } s^i = s^i_a \text{ and } f^i_s(q^i, s^i_r)! \end{cases}. \quad (3)$$

Hence, using (3), a transition labeled s^i_a is added in parallel with the transition labeled s^i_r to capture that the inserted event will lead the supervisor to the same state as the real event.

On the other hand, when event s^i_a is inserted by the attacker, the plant modeled by G^i can be at any state q^i_g for which $f^i(q^i_g, s^i_n)!$, where s^i_n is event such that $f^i_{sa}(f^i_{sa}(q^i_s, s^i_a), s^i_n)!$—i.e., event following s^i_a in S^i_a. To model the receptiveness of the plant to the attack, we add a loop with s^i_a to every state q^i_g in the G^i and generate the model of the physical plant under attack s^i_a, denoted by G^i_a. Automaton $G^i_a = (Q^i, E^i_{ga}, f^i_{ga}, q^i_0)$ where $E^i_{ga} = E^i \cup \{s^i_a\}$ and f^i_{ga} is defined as

$$f^i_{ga}(q^i, s^i) = \begin{cases} f^i_s(q^i, s^i), & \text{if } s^i \in E^i \text{ and } f^i(q^i, s^i)! \\ q^i, & \text{if } s^i = s^i_a \text{ and } f^i(q^i, s^i_n))! \\ & \text{and } f^i_{sa}(f^i_{sa}(q^i_s, s^i), s^i_n)! \end{cases}. \quad (4)$$

The second part of relation (4) models that the plant does not change the state on the attack event, but on the following event as imposed by the supervisor. Finally, the overall system behavior under attack G_A is captured by $G_A = (||_i S^i_a) \times (||_i G^i_a)$, where $S^i_a = S^i$ and $G^i_a = G^i$ if LC$_i$ is not under attack.

Running example continued: We illustrate the modeling of the insertion attack on communicating $b0$ between LC$_2$ and LC$_1$ in the running example—i.e., S^1 from Fig. 3(d) is attacked by inserting "fake" $b0a$. Using our approach, models of LC$_1$—S^1_a and cylinder A—G^1_a under attack are derived (see Fig. 5).

From S^1_a and G^1_a (see Fig. 5), as well as S^2 and G^2 (see Fig. 3), we obtain the model of the system under such attack—G_A (see Fig. 6); the states that G_A could enter after $b0$ insertion attack are marked red. The states in G_A are denoted by (x, y, z, u), where x, y, z, and u denote the state of S^1, S^2, G^1, and G^2, respectively. It can be observed that, to remain undetected at the moment of attack, the attack occurs when S^1 is at the state 1 or 4, which are receptive to $b0$. If the attack occurs while S^1 is at another state, it will be immediately detected.

To illustrate this, in Fig. 7, we provide the supervisors S^1_{adet} and S^2_{adet} with integrated state d that detects the receipt of an event at the state that is not receptive to this event. Namely,

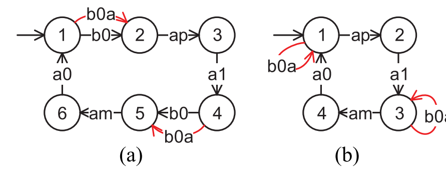

Fig. 5. Running example under $b0$ insertion attack on LC$_1$. (a) Automaton S^1_a—LC$_1$ under attack. (b) Automaton G^1_a—cylinder A under attack.

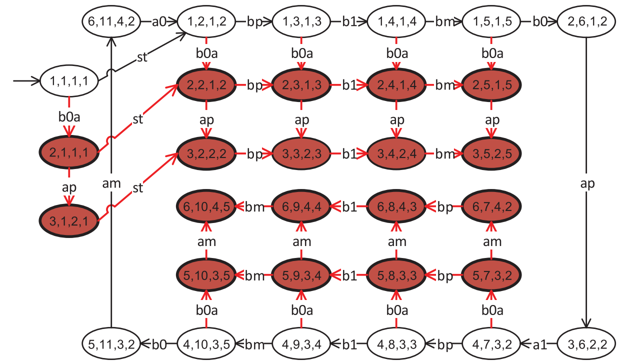

Fig. 6. Running example—$b0$ insertion attack on LC$_1$: (states that G_A can enter after attack are marked red).

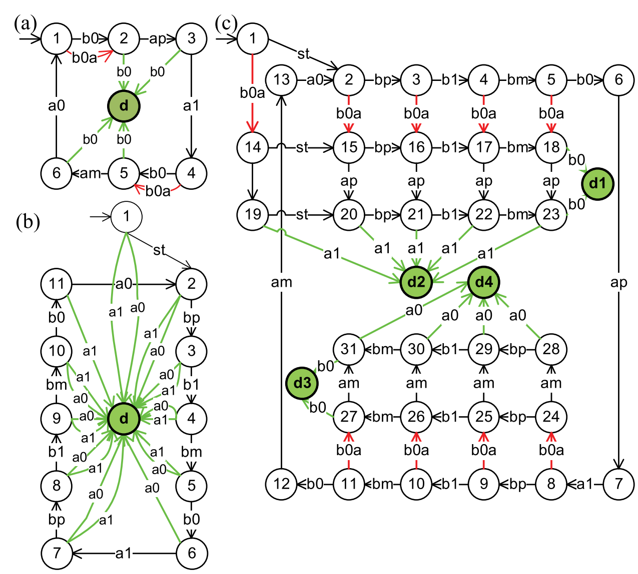

Fig. 7. Running example—$b0$ insertion attack on LC$_1$. (a) Automaton S^1_{adet} representing LC$_1$ under attack with integrated attack detection state d. (b) Automaton S^2_{adet} representing LC$_2$ with integrated attack detection state d. (c) $S_{adet} = S^1_{adet} || S^{2_{adet}}$.

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

S^1_{adet} [see Fig. 7(a)] is obtained from S^1_a [see Fig. 5(a)] by adding the detection state d and transitions labeled $b0$ from all the states not receptive to event $b0$ to d. Thus, automaton S^1_{adet} will enter the state d if it receives $b0$ at states other than 1 and 4. Similarly, automaton S^2_{adet} [see Fig. 7(b)] will enter state d if it receives $a0$ or $a1$ while at states not receptive to these events. Parallel operation of the supervisors S^1_{adet} and S^2_{adet} (i.e., $S_{adet} = S^1_{adet}||S^2_{adet}$) is presented in Fig. 7(c); in S_{adet}, $d1$–$d4$ correspond to the entrance of S^1_{adet} and/or S^2_{adet} in the state d. ■

Generally, S^i_{adet} can be derived from S^i_a as follows. $S^i_{adet} = (Q^i_{sdet}, E^i_{sa}, f^i_{sadet}, q^i_{0s})$ where $Q^i_{sdet} = Q^i_s \cup \{d\}$ and

$$f^i_{sadet}(q^i, s^i) = \begin{cases} f^i_{sa}(q^i, s^i), & \text{if } s^i \in E^i_{sa} \text{ and } f^i_{sa}(q^i, s^i)! \\ d, & \text{if } s^i \in E^i_{r_x} \\ & \text{and } \neg\, f^i_{sa}(q^i, s^i)! \end{cases}. \tag{5}$$

Implementing S^i_{adet} instead of S^i at LC$_i$ leads to immediate detection of any unexpected event that is received; this includes the insertion attack if it is not carried out while the supervisor is at the state that is receptive to the attack event. Conjoint operation of all S^i_{adet} in the system is presented as $S_{adet} = ||_i S^i_{adet}$, and it describes the conjoint behavior of all supervisors with integrated insertion attack detection implemented at LCs.

In addition to modeling a single insertion attack, a model of combined insertion attacks and the corresponding system behavior can be similarly obtained. Suppose that the supervisor S^i can be attacked by l_i different insertion attacks $s^i_{a_j}$, $j \in [1, \ldots, l_i]$. Following the presented procedure, all these attacks can be modeled by $S^i_{a_U}$, such that the language $L(S^i_{a_U}) = \cup_j L(S^i_{a_j})$, where $S^i_{a_j}$ is obtained applying relation from (3) for each of $s^i_{a_j}$. Similarly, we can obtain models of the plants under all insertion attacks $G^i_{a_U}$, as well as the model of the system under all insertion attacks G_{A_U}. Due to the properties of parallel composition, the language generated by the system under all insertion attacks modeled by G_{A_U} represents the union of languages generated by system under isolated attacks.

B. Removal Attack

Let us consider the removal attack that removes the event $c^i_{j,k} \in E^i_{r_x}$ that is sent to S^i from S^j; again, to simplify our notation, we use s^i_r to denote the “regular” event ($c^i_{j,k}$) and introduce event s^i_a to capture the attack. To remove event s^i_r, the adversary should attack when S^i is at a state q^i_r where $f^i_s(q^i_r, s^i_r)!$, while S^j is at one of the states $f^j_s(q^j, s^i_r)$. Furthermore, G^j should be in a state $f^j(q^j, s^i_r)$. As a result of the removal attack, S^i will remain at the state q^i_r, whereas the operation of S^j and G^j will continue as if attack did not occur. We capture the described system behavior as follows. The attack on s^i_r at state $q^i_r \in Q^i_s$ keeps S^i in q^i_r. Thus, LC$_i$ under attack can be modeled as automaton $S^i_a = (Q^i_s, E^i_{sa}, f^i_{sa}, q^i_{0s})$ where $E^i_{sa} = E^i_s \cup \{s^i_a\}$ and f^i_{sa} is expanded by adding self-loops on event s^i_a to the states q^i_r for which $f^i_s(q^i_r, s^i_r)!$—i.e.,

$$f^i_{sa}(q^i, s^i) = \begin{cases} f^i_s(q^i, s^i), & \text{if } s^i \in E^i_s \text{ and } f^i_s(q^i, s^i)! \\ q^i, & \text{if } s^i = s^i_a \text{ and } f^i_s(q^i, s^i_r)! \end{cases}. \tag{6}$$

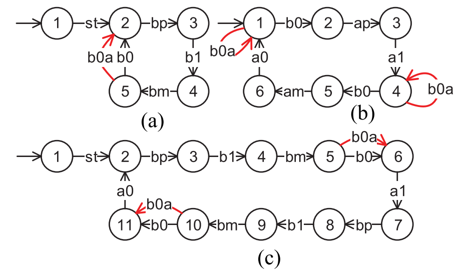

Fig. 8. Running example—$b0$ removal attack on LC$_1$. (a) Automaton G^2_a representing cylinder B under the attack. (b) Automaton S^1_a representing LC$_1$ under the attack. (c) Automaton S^2_a representing LC$_2$ under the attack.

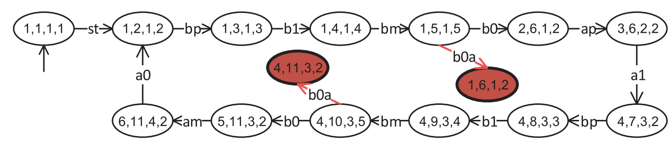

Fig. 9. Running example—$b0$ removal attack on LC$_1$: (states that G_A can enter after attack are marked red).

Since S^i_a does not change the state during attack, the attack will not influence the behavior of plant G^i.

To model the behavior of the transmitting module during the attack on the receiving controller, automata $S^j_a = (Q^j_s, E^j_{sa}, f^j_{sa}, q^j_{0s})$, with $E^j_{sa} = E^j_s \cup \{s^i_a\}$, and $G^j_a = (Q^j, E^j_{ga}, f^j_{ga}, q^j_0)$, with $E^j_{ga} = E^j \cup \{s^i_a\}$, are introduced. Note that by construction $s^i_r \in E^j_{t_x}$. The attack has no effect on S^j and G^j—i.e., they should continue their working cycle as if the attack did not occur. However, the attack affects operation of the overall system, which we model by f^j_{sa} and f^j_{ga} defined as

$$f^j_{sa}(q^j, s^j) = \begin{cases} f^j_s(q^j, s^j), & \text{if } s^j \in E^j_s \text{ and } f^j_s(q^j, s^j)! \\ f^j_s(q^j, s^i_a), & \text{if } s^j = s^i_a \text{ and } f^j_s(q^j, s^i_r)! \end{cases} \tag{7}$$

$$f^j_{ga}(q^j, s^j) = \begin{cases} f^j_s(q^j, s^j), & \text{if } s^j \in E^j \text{ and } f^j(q^j, s^j)! \\ f^j_s(q^j, s^i_a), & \text{if } s^j = s^i_a \text{ and } f^j(q^j, s^i_r)! \end{cases}. \tag{8}$$

Equations (7) and (8) add a transition labeled by s^i_a in parallel with the transition labeled by s^i_r to capture that LC$_j$ is not aware of the attack, and that it continues operation as if attack did not occur. The overall behavior of the system under attack G_A is now obtained as in the case of the insertion attack.

Running example continued: We illustrate the modeling of the removal attack on removal of $b0$ while transmitting it from LC$_2$ to LC$_1$, in our running example ($b0a$). Following the proposed modeling approach, models of LC$_1$ – S^1_a, LC$_2$ – S^2_a and cylinder B – G^2_a under attack are derived as presented in Fig. 8. Automaton G_A representing the behavior of the system under $b0$ removal attack is shown in Fig. 9—It can be observed that the removal of $b0$ leads to a deadlock, which stops the work-cycle, but will not lead to catastrophic damage. ■

V. Identification of Undesired System Behavior

In systems in which two-way communication between LCs exists—i.e., where S^i not only receives information from, but

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

Procedure 1: Identification of the Events Whose Communication Should be Protected.

INPUT:
$\Omega_c^k = \{w_{c,1}^k, \ldots, w_{c,l_k}^k\}$: set of l_k strings (events sequences) that would lead to catastrophic damage CD_k, $k \in 1, \ldots, M$
$s_{a_j}, j \in 1, \ldots, P$ possible attacks

```
for all attacks s_{a_j}, j = 1 to P do
    generate S_{a_j det}, G_{a_j}, G_{A_j det}, and Obs(G_{A_j det})
    for all CD_k k = 1 to M do
        for all strings that lead to CD_k, i = 1 to l_k do
            if Obs(G_{A_j det}) accepts w^k_{c,i} then
                s_r corresponding to s_{a_j} needs encryption
            end if
        end for
    end for
end for
```

also sends information to other LCs in the network, insertion attacks will be eventually revealed. Using our running example, this can be observed, e.g., in the case of S^1 that in regular operation 1) at state 1, receives $b0$ from S^2, while S^2 transits from state 5 to state 6, and 2) sends $a1$ to S^2 during the transition from state 3 to state 4, while S^2 is at state 6 [see Fig. 3(d) and (e)]. Now, let us assume that S^1 is attacked by $b0$ insertion attack—i.e., inserting event $b0a$—while at state 1; then, S^2 did not reach state 6. This is represented in Fig. 7(c) in states 1–5 corresponding to states $(1, y, 1, u)$ in G_A from Fig. 6. If S^2 reaches state 6 (and sends real $b0$) before S^1 reaches state 4, S^1 will receive real $b0$ while at state 2 or 3, and attack will be revealed; this is represented by transitions from states 18 and 23 to state $d1$ in Fig. 7(c). As an alternative, if S^1 comes into state 4 before S^2 enters state 6 (i.e., before it sends real $b0$), it will send $a0$ to S^2 that is not in the correct state and the attack will be revealed again as illustrated on transitions from states 19–23 to $d2$ in Fig. 7(c). Thus, the attack is detected at one of the states corresponding to G_A states $(3, y, 2, u)$, $(x, 5, z, 5)$ (see Fig. 6). To summarize, by implementing S^i_{adet} instead of S^i at LCs, the insertion attacks will be detected at some point for systems in which two-way communication is present.

Note that when LCs have, both, sensor and actuator signals, two-way communication is always present. On the other hand, if only actuators or sensors are mapped to LC, two-way communication is introduced with acknowledgment signals used for safety reasons. Thus, the attack will be detected at some point. Nevertheless, between attack occurrence and detection, in general, the system will not behave as desired. The question is whether the system behavior after attack will lead to significant damage, e.g., to the collision of systems' elements or manufactured parts damage.

System behaviors that lead to catastrophic damage CD_k, $k \in 1, \ldots, M$ can be described by a set of undesired event strings $\Omega_c^k = \{w_{c,1}^k, \ldots, w_{c,l_k}^k\}$. The question is whether the system will exhibit a sequence from Ω_c^k, i.e., will CD_k occur under attack event s_a^i before the attack is revealed. Namely, if s_a^i potentially leads to CD_k, then communication of s_r^i between LCs has to be protected. To answer the question, we employ the automaton G_{Adet} that represents the system behavior under the attack event s_a^i. This automaton incorporates states for detection of the event being received at a wrong state, and it is obtained from S_{adet} and G_a, as $G_{Adet} = S_{adet} \times G_a$.

Here, G_{Adet} contains the unobservable event[4] s_a^i that will break the chain of events from Ω_c^k and cannot be directly used for checking whether the system will exhibit the behavior specified by Ω_c^k, since the strings from Ω_c^k do not contain s_a^i. Event s_a^i could be easily eliminated from G_{Adet} by a natural projection. However, this could lead to generation of a nondeterministic automaton; to solve this issue and to preserve language equivalence, observer $Obs(G_{Adet})$ of G_{Adet} should be generated [37]. If $Obs(G_{Adet})$ accepts any string from Ω_c^k, than CD_k could happen during the s_a^i attack, and communication of s_r^i should be protected, as summarized in Procedure 1. It should be noted that $Obs(G_{Adet})$ is used offline, during system design, to model the behavior of the system under attack and to identify communication channels that require protection. The observer that considers all insertion attacks s_{a_j} simultaneously, is obtained from $G_{A_U det}$, and the language $L(Obs(G_{A_U det}))$ represents the union of languages $L(Obs(G_{A_j det}))$.

We illustrate the use of Procedure 1 on our running example.

Running example continued: In the running example, three insertion and three removal attacks could occur—$a0$ and $a1$ on communication from LC_1 to LC_2, and $b0$ on communication from LC_2 to LC_1. The regular cycle of the system can be presented by string $w_r \in \Sigma_r$, where $\Sigma_r = \{st(bp\ b1\ bm\ b0\ ap\ a1\ bp\ b1\ bm\ b0\ am\ a0)^*,\ st\}$. Mechanical design of the system [see Fig. 1(a)] is such that marker can come into the position I to take marking liquid and leave it either in the horizontal or in the vertical direction (note that in regular work-cycle approaching and leaving are in vertical direction). On the other hand, it can enter and leave position II only in the vertical direction; otherwise the marking liquid could be diffused over the part thus endangering marking quality. Furthermore, to ensure part marking, it is necessary that cylinder B reaches end position before retracting at both, positions I and II. Thus, there exist three situations that endanger the quality of the process: 1) CD_1—marker enters position II from horizontal direction, 2) CD_2—marker leaves position II in horizontal direction, and 3) CD_3—cylinder B retracts before reaching end position. For each of these situations, events strings sets Ω_c^k can be identified as presented in Table II.

$Obs(G_{Adet})$ that contains all possible consequences of insertion attacks on $b0$ is presented in Fig. 10. It is obtained from $G_{Adet} = S_{adet} \times G_a$, where $S_{adet} = S^1_{adet} || S^2_{adet}$ (see Fig. 7) and $G_a = G_a^1 || G^2$ [see Fig. 5(b) and 3(b)]; states $d1$–$d11$ are derived from states d in S^1_{adet} and/or S^2_{adet}. In case of $b0$ insertion attack, CD_2 could occur (strings $w^2_{c,1}$, $w^2_{c,2}$, and $w^2_{c,3}$). Fig. 11 represents $Obs(G_{A_U det})$; it can be observed that in the case of $a1$ insertion attack, the occurrence of CD_1 is possible (string $w^1_{c,1}$). Nevertheless, $a0$ insertion attack will not have

[4]By design, the system is not aware that attack signals are attack; thus these are not observable events.

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

TABLE II
RUNNING EXAMPLE: $\Omega_c{}^k$ DEFINITION

CD_1	$w^1_{c,1} = w_{r1}(ama0ap)^*bp$, $w^1_{c,2} = w_{r1}(amap)^*bp$ where $w_{r1} = w_r bpb1bmb0ap(a1ama0ap)^*$
CD_2	$w^2_{c,1} = w_{r2}am$, $w^2_{c,2} = w_{r2}b1am$, $w^2_{c,3} = w_{r2}b1bmam$ where $w_{r2} = w_r bpb1bmb0apa1bp(b1bmb0bp)^*$
CD_3	$w^3_{c,1} = w_{r3}bm$, $w^3_{c,2} = w_{r3}b1bmb0apa1bpbm$ where $w_{r3} = w_r bp$

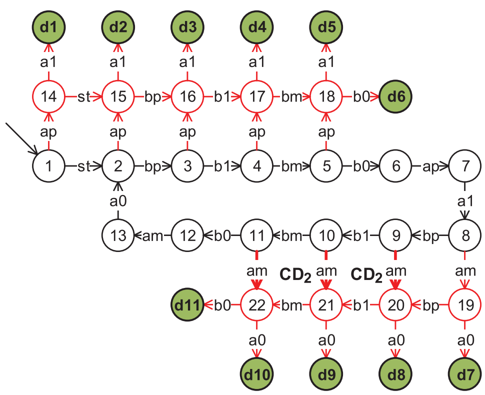

Fig. 10. Running example—$b0$ insertion attack: Automaton $Obs(G_{Adet})$ that represents all possible system behaviors under $b0$ insertion attack.

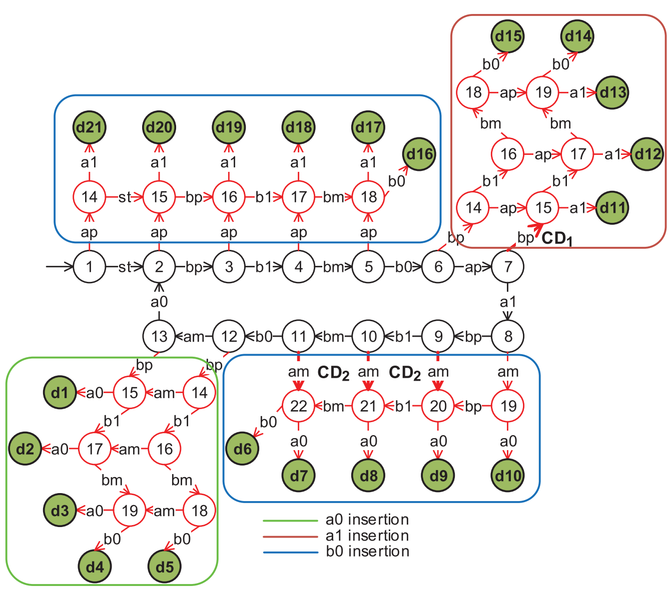

Fig. 11. Running example—$Obs(G_{AUdet})$ that represents all possible system behaviors under $a0$, $a1$, and/or $b0$ insertion attacks.

catastrophic effect on the system performance. Furthermore, neither of insertion attacks would cause CD_3. All three removal attacks will lead to deadlock, as presented in Fig. 9 for $b0$ removal attack. System behavior models are similar in the case of $a0$ or $a1$ removal attacks. Thus, removal attacks will lead to none of CD_k. Consequently, the communication of $b0$ and $a1$ should be protected, whereas for $a0$ encryption is not necessary. ∎

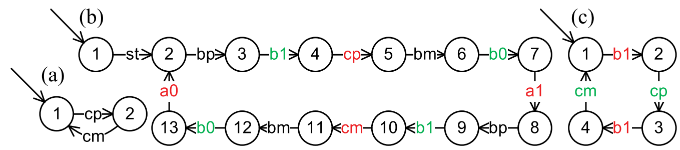

Fig. 12. Case study. (a) Automaton G^3 representing gripper C. (b) Automaton S^2 representing LC_2. (c) Automaton S^3 representing LC_3.

VI. INDUSTRIAL CASE STUDY

We consider a case study that refers to the manipulator obtained by reconfiguring the marking device from our running example, as presented in Fig. 1(b). We also considered a more complex system with concurrent processes (specifically, the case study given in [31]), and similar results were obtained. Due to the space constraint, the detailed system analysis for the second case study has been omitted from this work.

The manipulator has two translational degrees of freedom realized by smart cylinders A and B as in running example (see Table I). It is also equipped with a smart vacuum gripper C that is controlled by a monostable dual control valve 3/2, and has integrated LC_3 with mapped signal cp for part gripping and cm for part releasing. Manipulator moves elastic part from positions I to II and performs the following work cycle:

$$B+\ C+\ B-\ A+\ B+\ C-\ B-\ A- \quad (9)$$

where cylinder activities are denoted as in (1), whereas $C+$ refers to part gripping and $C-$ to part releasing. Work cycle is started by pressing start switch (st) mapped to LC_2.

A. Attack Modeling and Identification of Undesired System Behaviors

Following the introduced modeling approach, automata G^1, G^2, and G^3 representing all possible legal behaviors of cylinders A and B, and gripper C are generated. G^1 and G^2 are the same as in the running example [see Fig. 3(a) and (b)], whereas G^3 has the following set of events $E^3 = E^3_o = E^3_c = \{cp, cm\}$ and it is shown in Fig. 12(a). Local controllers LC_1, LC_2, and LC_3 impose supervisors S^1, S^2, and S^3, respectively. Supervisor S^1 is the same as in the running example [see Fig. 3(d)] and has event set $E^1_s = \{ap, a1, am, a0, b0\}$ with the event $b0 = c^1_{2,1}$ that is communicated from LC_2. Supervisor S^2 [see Fig. 12(b)] is based on the following events set $E^2_s = \{st, bp, b1, bm, b0, a0, a1, cp, cm\}$ and it has four communicated events: 1) $a0 = c^2_{1,1}$ and $a1 = c^2_{1,2}$ received from LC_1, and 2) $cp = c^2_{3,1}$ and $cm = c^2_{3,2}$ received from LC_3. Finally, S^3 [see Fig. 12(c)] contains the events $E^3_s = \{cp, cm, b1\}$, where $b1 = c^3_{2,1}$ is received from LC_2. Note that gripper C does not contain sensors and that acknowledgment events cp and cm are sent from S^3 to S^2 for safety reasons to ensure two-way communication in S^3 as elaborated in Section V.

The strings $w_r \in \Sigma_r$, where $\Sigma_r = \{st(bp\ \ b1\ \ cp\ \ bm\ \ b0\ ap\ a1\ bp\ b1\ cm\ bm\ b0\ am\ a0)^*,\ st\}$ define regular cycles of the system. On the other hand, catastrophic damages could

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

TABLE III
CASE STUDY: Ω_c^k DEFINITION

CD_1	$w_{c,1}^1 = w_{r1}ap$, $w_{c,2}^1 = w_{r1}bmap$, $w_{c,3}^1 = w_{r1}bmb0bpap$ where $w_{r1} = w_r bpb1cp(bmb0bpb1)^*$
CD_2	$w_{c,1}^2 = w_{r2}am$, $w_{c,2}^2 = w_{r2}b1am$, $w_{c,3}^2 = w_{r2}b1bmam$ where $w_{r2} = w_r bpb1cpbmb0apa1bp(b1bmb0bp)^*$
CD_3	$w_{c,1}^3 = w_{r3}(ama0ap)bp$, $w_{c,2}^3 = w_{r3}(amap)^* bp$ where $w_{r3} = w_r bpb1cpbmb0ap(a1ama0ap)^*$
CD_4	$w_{c,1}^4 = w_{r4}bp$ where $w_{r4} = w_r bpb1cpbmb0apa1(bpb1bmb0)^* am$

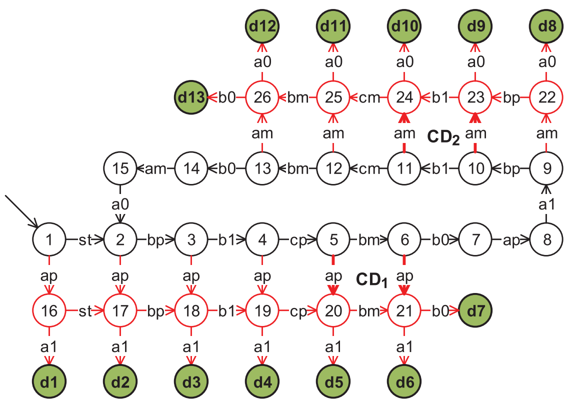

Fig. 13. Case study—$Obs(G_{Adet})$: $b0$ insertion attack on LC_1.

happen in the following scenarios (described by events string sets $w_{c,lk}^k$, $k \in \{1, 2, 3, 4\}$ defined in Table III):

1) CD_1 manipulator with gripped part in position I tries to advance cylinder A before retracting cylinder B;
2) CD_2 manipulator in position II tries to retract A before releasing part and before retracting B;
3) CD_3 manipulator tries to put down the part while moving it from position I to position II;
4) CD_4 manipulator does not leave the part in position II and tries to put it down while moving it from position II to position I.

Observers for insertion attacks: 1) $b0$ and 2) $a1$ and $a0$ are shown in Fig. 13 and 14, respectively. From these figures, it can be observed that CD_1 can appear during $b0$ insertion attack on LC_1 ($w_{c,1}^1$ and $w_{c,2}^1$ on transitions from 5 to 20 and 6 to 21—Fig. 13) and during $a1$ insertion attack on LC_2 [$w_{c,3}^1$ and $w_{c,1}^1$ on transitions from 16 to 17 and 18 to 19—Fig. 14(a)]. Furthermore, CD_2 can occur during $b0$ insertion attack on LC_1 ($w_{c,1}^2$ and $w_{c,2}^2$ on transitions from 10 to 23 and 11 to 24—Fig. 13). Other insertion attacks (observers are omitted due to space limitation) will not lead to CDs. Furthermore, removal attacks will lead to immediate deadlock and will not cause any damage. Thus, transmissions of $b0$ from LC_2 to LC_1 and of $a1$ from LC_1 to LC_2 should be protected.

B. Experimental Validation

We experimentally evaluated our approach to attack modeling and detection on a real-world industrial case-study—industrial

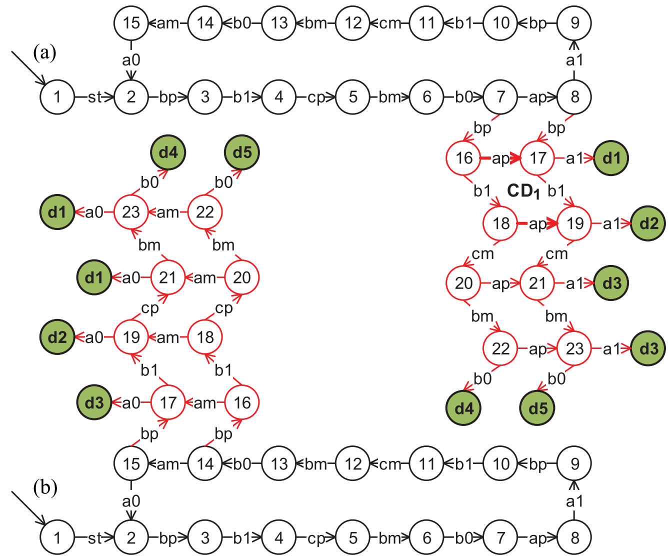

Fig. 14. Case study—$Obs(G_{Adet})$. (a) $a1$ and (b) $a0$ insertion attack on LC_2.

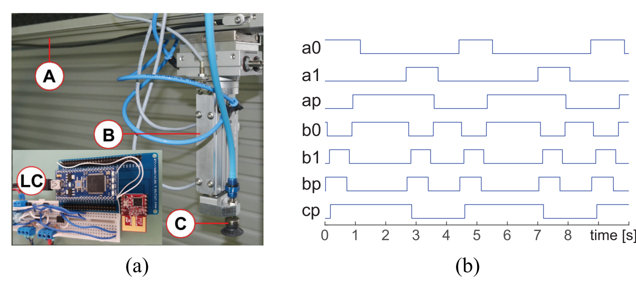

Fig. 15. Industrial case study. (a) Experimental installation. (b) Timing diagram capturing inputs and outputs of the system in a scenario without attack.

manipulator shown in Fig. 15. Each actuator (two cylinders and gripper) represents a smart device with its own LC, where the mapping of sensors and actuators is captured in Table I for cylinders A and B while cp and cm are mapped to LC_3 (i.e., the gripper's LC).

Hence, the control system of the manipulator from Fig. 15 is implemented using three wireless nodes (LCs); we employed ARM Cortex-M3 microcontroller boards that communicate over IEEE 802.15.4-compliant wireless transceivers. LC_1–LC_3 implement S_{adet}^1, S_{adet}^2, and S_{adet}^3 obtained from S^1 [shown in Fig. 3(d)], S^2 [see Fig. 12(b)], and S^3 [see Fig. 12(c)] by adding the attack detection state. On the entrance to the state d at any of the LCs, the system stops immediately. Timing diagram capturing the sequence of controllable and uncontrollable events acquired from the real-world manipulator during a regular work-cycle (i.e., without attack) is presented in Fig. 15; due to space constrains, we do not show am, bm, and cm signals, as they are only inverted signals of the ap, bp, and cp, respectively.

In addition to wireless nodes implementing the distributed controllers, the experimental installation also contains the fourth LC, based on the same ARM board, which is used as an attacker. The attacker is completely aware of the system design

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

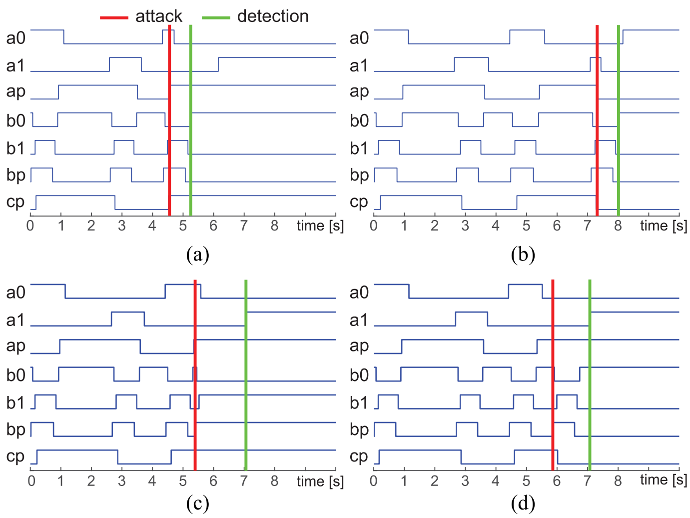

Fig. 16. Case study. Experimentally captured timing diagrams of the inputs and outputs of the manipulator in the presence of (a) and (b) bo insertion attacks and (c) and (d) $a1$ insertion attacks at different time instants.

and performance, as it can eavesdrop all communication between control LCs and has the knowledge of the LCs design. Thus, the attacker is capable of crafting attacks that will not be immediately revealed. To validate the proposed method for attacks modeling and detection, using the attack LC, we have implemented all attacks on the manipulator from the case-study (i.e., Section VI-A). The observed executions of the system were completely compliant with the previously described observers (in Fig. 13 and 14).

For example, in Fig. 16, we present the timing diagrams experimentally acquired from the system in the presence of attacks; specifically, we illustrate system performance under $b0$ and $a1$ insertion attacks launched at different time instants; note that the attacks may have different impact based on the timing instance in which they are launched. We first illustrate system execution under the $b0$ insertion attack that was activated at time $t = 4.55$ s, resulting in the sequence $w_r\ bp\ b1\ cp\ \mathbf{ap}\ bm\ b0$,[5] as shown in Fig. 16(a), which corresponds to transitions through states 5, 20, 21, and $d7$ from Fig. 13 before attack detection at $d7$; this attack leads to CD_1.

On the other hand, $b0$ insertion attack at time $t = 7.31$s results in the event sequence, shown in Fig. 16(b): $w_r\ bp\ b1\ cp\ bm\ b0\ ap\ a1\ bp\ b1\ cm\ \mathbf{am}\ bm\ b0$; this reflects transitions through states 12, 25, and 26 before detection at $d13$ (see Fig. 13) and does not lead to any catastrophic damage.

Similarly, $a1$ insertion attacks at times $t = 5.40$ s and $t = 5.86$ s lead to the event sequences from Fig. 16(c) and (d): 1) $w_r\ bp\ b1\ cp\ bm\ b0\ ap\ \mathbf{bp}\ b1\ a1$ corresponding to transitions 8, 17, 19, and $d2$ in the observer from Fig. 14(a), and 2) $w_r\ bp\ b1\ cp\ bm\ b0\ ap\ \mathbf{bp}\ b1\ cm\ bm\ b0\ a1$ corresponding to the transitions 8, 17, 19, 21, 23, and $d5$ in the observer from Fig. 14(a) before attack detection. Neither of the illustrated $a1$ insertion attacks leads to catastrophic damage of the system.

VII. Conclusion

In this article, we focused on security challenges in the design of sequential control systems for industrial automation, where the control was distributed over IIoT-enabled smart devices. We presented a method for modeling relevant attacks on communication between such LCs, which share information about local events to ensure their coordination and the desired overall system operation. We focused on event-insertion and event-removal attacks that allow us to capture a wide-range of standard attacks on industrial systems, such as denial-of-service attacks, false-data injection attacks, as well as man-in-the-middle attacks. We considered attacks that cannot be immediately detected, in order to have significant impact on system operation, and for such attacks, we presented methods to model their impact on the system. To achieve this, we employed a standard SCT framework that is widely adopted for modeling of sequential control systems used for industrial automation; this allows for modeling of both physical behavior of smart IIoT-enabled devices as well as cyber behavior of their LCs in the presence of the attacks.

In the considered case studies, we showed that stealthy event-removal attacks lead to system deadlock; the reason is that the considered systems for safety reasons already employ two-way communication where every command is followed by either a corresponding sensing event or a communication acknowledgment event. The deadlock is immediate for such systems that do not have parallel (concurrent) processes, since in these processes, there is no branching in an input and output sequence, and removing any actuation command or sensing event would prevent continuation of the system execution.

On the other hand, in such systems (i.e., with such two-way communication) that do contain parallel processes, the deadlock is immediate on the attacked branch, whereas the parallel branches continue work-cycle until they converge with the attacked branch; the deadlock on the whole system occurs at the convergence point.[6] It should be noted that concurrent processes in sequential control are parallel in their nature and do not impose any time-related, mechanical, or other constraints on branch parallelism that could lead to security related issues.

Furthermore, we showed that due to two-way communication between LCs, event-insertion attack can be eventually revealed using the developed detection mechanism. Nevertheless, between attack occurrence and detection, the system can exhibit undesired behaviors that could result in significant damage. Hence, we provided a method to identify events whose communication should be protected, to ensure satisfiable system operation in resource-constrained systems, in the presence of attacks.

The proposed method was experimentally verified using a real-world case study with three LCs. For the systems with higher number of LCs and with higher complexity of control

[5]Under $b0$ insertion attacks, the occurrence of uncontrollable events $a1$ or $a0$ after attack detection is caused by the controllable events ap/am and system inertia.

[6]As captured in Section VI, we have also performed security analysis for a system with parallel processes. However, due to the space constraints, a detailed description of the example of such system is omitted from this article.

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

tasks where a large number of commands were executed between subsequent communications, the sequence of events can be modeled by higher level of abstraction, such as macrosteps in Grafcet [38]. In this way, a hierarchical structure can be introduced into events. Our future efforts will include timing-based analysis of the system under attacks, and the use of (time) intermittent authentication to protect communication.

Acknowledgment

This article was submitted for publication prior to Dr. Lesi joining Intel Labs, and the views expressed do not necessarily reflect those of Intel Corporation. The authors would like to express the gratitude to SMC Industrial Automation Serbia for providing experimental equipment.

References

[1] H. Kagermann, W. Wahlster, and J. Helbig, *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0.*, 2013. [Online]. Available: http://www.acatech.de
[2] H. ElMaraghy *et al.*, “Product variety management,” *CIRP Ann. Manuf. Technol.*, vol. 62, no. 2, pp. 629–652, 2013.
[3] Y. Koren, X. Gu, and W. Guo, “Reconfigurable manufacturing systems: Principles, design, and future trends,” *Frontiers Mech. Eng.*, vol. 13, no. 2, pp. 121–136, 2018.
[4] V. Lesi, Z. Jakovljevic, and M. Pajic, “Towards plug-n-play numerical control for reconfigurable manufacturing systems,” in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2016, pp. 1–8.
[5] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, “The industrial Internet of things (IIOT): An analysis framework,” *Comput. Ind.*, vol. 101, pp. 1–12, 2018.
[6] M. Cheminod, L. Durante, and A. Valenzano, “Review of security issues in industrial networks,” *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 277–293, Feb. 2013.
[7] IIC, *The Industrial Internet of Things Volume G1: Reference Architecture*, 2017. [Online]. Available: https://www.iiconsortium.org
[8] V. Lesi, I. Jovanov, and M. Pajic, “Security-aware scheduling of embedded control tasks,” *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, 2017, Art. no. 188.
[9] S. Goel and R. Negi, “Guaranteeing secrecy using artificial noise,” *IEEE Trans. Wireless Commun.*, vol. 7, no. 6, pp. 2180–2189, Jun. 2008.
[10] Y. Zou and G. Wang, “Intercept behavior analysis of industrial wireless sensor networks in the presence of eavesdropping attack,” *IEEE Trans. Ind. Informat.*, vol. 12, no. 2, pp. 780–787, Apr. 2016.
[11] L. Zhou, K. H. Yeh, G. Hancke, Z. Liu, and C. Su, “Security and privacy for the industrial Internet of things: An overview of approaches to safeguarding endpoints,” *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 76–87, Sep. 2018.
[12] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, “A secure control framework for resource-limited adversaries,” *Automatica*, vol. 51, pp. 135–148, 2015.
[13] A. O. De Sá, L. F. R. da Costa Carmo, and R. C. S. Machado, “Covert attacks in cyber-physical control systems,” *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1641–1651, Aug. 2017.
[14] L. 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.
[15] P. M. Lima, M. V. S. Alves, L. K. Carvalho, and M. V. Moreira, “Security against network attacks in supervisory control systems,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12333–12338, 2017.
[16] R. Su, “Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations,” *Automatica*, vol. 94, pp. 35–44, 2018.
[17] M. Wakaiki, P. Tabuada, and J. P. Hespanha, “Supervisory control of discrete-event systems under attacks,” *Dyn. Games Appl.*, vol. 9, no. 4, pp. 965–983, 2019.
[18] R. Fritz and P. Zhang, “Modeling and detection of cyber attacks on discrete event systems,” *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 285–290, 2018.
[19] Q. Zhang, Z. Li, C. Seatzu, and A. Giua, “Stealthy attacks for partially-observed discrete event systems,” in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2018, pp. 1161–1164.
[20] R. M. Ges, E. Kang, R. Kwong, and S. Lafortune, “Stealthy deception attacks for cyber-physical systems,” in *Proc. IEEE 56th Annu. Conf. Decision Control*, 2018, pp. 4224–4230.
[21] Y. Wang, A. K. Bozkurt, and M. Pajic, “Attack-resilient supervisory control of discrete-event systems,” 2019, *arXiv:1904.03264*. [Online]. Available: http://arxiv.org/abs/1904.03264
[22] P. Lima, M. Alves, L. Carvalho, and M. Moreira, “Security against communication network attacks of cyber-physical systems,” *J. Control, Autom. Elect. Syst.*, vol. 30, no. 1, pp. 125–135, 2019.
[23] R. Fritz, P. Schwarz, and P. Zhang, “Modeling of cyber attacks and a time guard detection for ICS based on discrete event systems,” in *Proc. Eur. Control Conf.*, 2019, pp. 4368–4373.
[24] A. Rashidinejad, B. Wetzels, M. Reniers, L. Lin, Y. Zhu, and R. Su, “Supervisory control of discrete-event systems under attacks: An overview and outlook,” in *Proc. Eur. Control Conf.*, 2019, pp. 1732–1739.
[25] P. Ramadge and W. M. Wonham, “The control of discrete event systems,” *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.
[26] P. M. Lima, L. K. Carvalho, and M. V. Moreira, “Detectable and undetectable network attack security of cyber-physical systems,” *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 179–185, 2018.
[27] M. Pajic *et al.*, “Robustness of attack-resilient state estimators,” in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, 2014, pp. 163–174.
[28] Y. Mo, R. Chabukswar, and B. Sinopoli, “Detecting integrity attacks on SCADA systems,” *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 4, pp. 1396–1407, Jul. 2014.
[29] Y. Mo *et al.*, “Cyber–physical security of a smart grid infrastructure,” *Proc. IEEE*, vol. 100, no. 1, pp. 195–209, Jan. 2012.
[30] L. Ricker, S. Lafortune, and S. Genc, “DESUMA: A tool integrating GIDDES and UMDES,” in *Proc. 8th Int. Workshop Discrete Event Syst.*, 2006, pp. 392–393.
[31] Z. Jakovljevic, V. Lesi, S. Mitrovic, and M. Pajic, “Distributing sequential control for manufacturing automation systems,” *IEEE Trans. Control Syst. Technol.*, to be published.
[32] Y. Qamsane, A. Tajer, and A. Philippot, “A synthesis approach to distributed supervisory control design for manufacturing systems with Grafcet implementation,” *Int. J. Prod. Res.*, vol. 55, no. 15, pp. 4283–4303, 2017.
[33] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets,* 2nd ed. Berlin, Germany: Springer, 2010.
[34] F. Schneider, “Enforceable security policies,” in *Proc. Found. Intrusion Tolerant Syst.*, 2003, pp. 117–137.
[35] M. Droste and R. Shortt, “From petri nets to automata with concurrency,” *Appl. Categorical Struct.*, vol. 10, no. 2, pp. 173–191, 2002.
[36] M. Cantarelli and J.-M. Roussel, “Reactive control system design using the supervisory control theory: Evaluation of possibilities and limits,” in *Proc. 9th Int. Workshop DES*, 2008, pp. 200–205.
[37] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Berlin, Germany: Springer, 2008.
[38] *IEC 60848:2013 GRAFCET Specification Language for Sequential Function Charts*, Geneva, Switzerland, 2013.

Zivana Jakovljevic (Member, IEEE) received the Dipl. Ing., M.Sc., and Ph.D. degrees in mechanical engineering from the Faculty of Mechanical Engineering, University of Belgrade, Belgrade, Serbia, in 1999, 2004, and 2010, respectively.

She is currently an Associate Professor and the Head of the Laboratory for Manufacturing Automation, Faculty of Mechanical Engineering, University of Belgrade, where she has been a Member of Academic Staff since 2001. Her current research interests include intelligent manufacturing systems, cyber-physical systems, industrial Internet of Things, distributed control, 3-D vision systems in manufacturing automation, machine learning, and nonstationary signal processing.

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.

Vuk Lesi received the B.Sc. degree from the University of Belgrade, Belgrade, Serbia, in 2015, and the Ph.D. degree from Duke University, Durham, NC, USA, in 2019, both in electrical and computer engineering.

He is currently a Research Scientist with Security and Privacy Research, Intel Labs, Hillsboro, OR, USA. His personal research interests include design and analysis of reconfigurable manufacturing systems, distributed industrial automation, embedded and real-time control, and security-aware safety-critical cyber-physical systems.

Dr. Lesi was the recipient of multiple recognitions including the Best Paper Award at the 2017 ACM SIGBED International Conference on Embedded Software.

Miroslav Pajic (Senior Member, IEEE) received the Dipl.Ing. and M.S. degrees from the University of Belgrade, Belgrade, Serbia, in 2003 and 2007, respectively, and the M.S. and Ph.D. degrees from the University of Pennsylvania, Philadelphia, PA, USA, in 2010 and 2012, respectively, all in electrical engineering.

He is currently the Nortel Networks Assistant Professor with the Electrical and Computer Engineering Department, Duke University, Durham, NC, USA, with a secondary appointment with the Computer Science Department. His current research focuses on the design and analysis of high-assurance cyber-physical systems with varying levels of autonomy and human interaction.

Dr. Pajic was the recipient of various awards including the ACM SIGBED Early-Career Award, IEEE TCCPS Early-Career Award, NSF CAREER Award, ONR Young Investigator Award, ACM SIGBED Frank Anger Memorial Award, Joseph and Rosaline Wolf Best Dissertation Award from Penn Engineering, IBM Faculty Award, as well as seven Best Paper and Runner-up Awards. He is an Associate Editor for the *ACM Transactions on Computing for Healthcare* and a Co-Chair of the 2019 ACM/IEEE International Conference on Cyber–Physical Systems.

Authorized licensed use limited to: Duke University. Downloaded on March 10,2021 at 16:23:50 UTC from IEEE Xplore. Restrictions apply.