Security Analysis for Distributed IoT-Based Industrial Automation

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

***Abstract*—Internet of Things (IoT) technologies enable development of Reconfigurable Manufacturing Systems—a new generation of modularized industrial equipment suitable for highly-customized manufacturing. Sequential control in these systems is largely based on discrete events, whereas their formal execution semantics is specified as Control Interpreted Petri Nets (CIPN). Despite industry-wide use of programming languages based on the CIPN formalism, formal verification of such control applications in the presence of adversarial activity is not supported. Consequently, in this paper we introduce security-aware modeling and verification techniques for CIPN-based sequential control applications. Specifically, we show how CIPN models of networked industrial IoT controllers can be transformed into Time Petri Net (TPN)-based models, and composed with plant and security-aware channel models in order to enable system-level verification of safety properties in the presence of network-based attacks. Additionally, we introduce realistic channel-specific attack models that capture adversarial behavior using nondeterminism. Moreover, we show how verification results can be utilized to introduce security patches and facilitate design of attack detectors that improve system resiliency, and enable satisfaction of critical safety properties. Finally, we evaluate our framework on an industrial case study.**

***Note to Practitioners*—Our main goal is to provide formal security guarantees for distributed sequential controllers. Specifically, we target smart automation controllers geared towards Industrial IoT applications, that are typically programmed in C/C++, and are running applications originally designed in e.g., GRAFCET (IEC 60848)/SFC (IEC 61131-3) automation programming languages. Since existing tools for design of distributed automation do not support system-level verification of relevant safety properties, we show how security-aware transceiver and communication models can be developed and composed with distributed controller models. Then, we show how existing tools for verification of Time Petri Nets can be used to verify relevant properties including safety and liveness of the distributed automation system in the presence of network-based attacks. To provide an end-to-end analysis as well as security patching, results of our analysis can be used to deploy suitable firmware updates during the stage when executable code for target controllers (e.g., in C/C++) is generated based on GRAFCET/SFC control models. We also show that security guarantees can be improved as the relevant safety/liveness properties can be verified after corresponding security patches are deployed. Finally, we show applicability of our framework on a realistic distributed pneumatic manipulator.**

***Primary and Secondary Keywords*—Primary Topics: Sequential control systems, Secure distributed automation, Industrial Internet of Things; Secondary Topics: Petri nets, Non-deterministic analysis.**

V. Lesi and M. Pajic are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, 27708 USA (email: vuk.lesi@duke.edu; miroslav.pajic@duke.edu.

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

Manuscript received December 15, 2019; revised August 20, 2020, April 2, 2021, and July 19, 2021.; accepted August 15, 2021.

I. Introduction

Advanced capabilities of smart Internet of Things (IoT) devices have lead to their widespread adoption in industrial automation system, rapidly advancing reconfigurable manufacturing [1]; the rise of the fourth industrial revolution, known as Industry 4.0 [2], introduces the new era of highly-customized (rather than highly-serialized) manufacturing [3], [4]. In this vision, manufacturing resources are highly modularized, providing the necessary flexibility to adapt to dynamical market demands [5]. Efficient structural and functional changes are supported by Reconfigurable Manufacturing Systems (RMS) that can be configured *ad-hoc* with little or zero downtime [6].

The foundation of RMS are modules controlled by smart Industrial IoT (IIoT)-enabled controllers. IIoT endpoints (sometimes referred to as *industrial assets*) are heterogeneous by definition—they represent multi-vendor components whose deployment environment dynamically changes depending on the process needs and current configuration of RMS. Also, a plethora of communication technologies (wired and wireless) and protocols are employed [7] for interaction between assets, whereas Radio Frequency Identification (RFID) is the technology of choice for their linking with process flow data [8]. Seamless reconfiguration, integration and reliable functioning of RMS requires that components are highly autonomous. Specifically, they must be capable of seamlessly communicating with each other using compatible protocols (*integrability*), exchanging both low-level control-related and high-level process-bound information (*interoperability*), and to interact with each other in different ways to support a variety of configurations (composability) [9].

Reconfigurability is naturally supported by distributed control architectures [10]. Conventionally centralized controllers are responsible of all aspects of control—from low-level event signaling to high-level coordination; yet, their complexity hinders reconfigurability both from the hardware perspective (e.g., requiring component re-wiring), and the software aspect (e.g., having to ensure the control software is aware of and functions correctly under the new hardware configuration) [11], [12]. Thus, the new generation of smart manufacturing resources must exploit not only functionally-required components (such as sensors and actuators) but also intrinsic computation and communication capabilities of IIoT-enabled controllers in order to enable a higher level of automation and autonomy [13], [14]. Control distribution enables decoupling of fine-grained

details about *how* control over the specific physical resource is performed, from the resources coordination problem that only needs to worry about *what* the manufacturing resources are capable of performing.

The networked nature of the new generation of distributed automation systems makes them susceptible to network-based attacks [15], similar to security vulnerabilities reported in other cyber-physical systems domains (e.g., [16]). For example, an adversary may inject false events [17], delay or deny network access to legitimate controllers [18], or manipulate control commands [19] sent over unsecure communication channels. To illustrate this, consider the distributed automation system from Fig. 2(a), which we will use as a running example in this work, consisting of conveyor belts and a pick & place station; adversarial actions over unprotected communication between conveyor and pick & place controllers may cause a stall in forwarding workpieces from belts 1/2 to the belt 3, or even collision between pick & place actuators and workpieces (e.g., when workpiece presence signals are falsified).

Providing security and safety guarantees is critical in distributed sequential control systems directly impacted by communication unavailability. Yet, despite devastating effects such attacks may have on operation of distributed industrial automation systems, existing approaches to securing such systems are somewhat *ad-hoc*; e.g., impromptu use of packet encryption for confidentiality or authentication for integrity, especially since real-time encryption algorithms efficient enough for industrial applications are scarce [20]. Commonly, the impact of the employed security mechanisms on safety and control performance (i.e., Quality-of-Control—QoC) are unclear and hard to evaluate if no formal system analysis can be performed.

Consequently, to enable building of secure and correct-by-design RMS, in this work we introduce efficient techniques for systematic security analysis of distributed control applications deployed on IIoT-enabled local controllers (LCs). We show how results of the security analysis can be used to improve QoC and safety guarantees in the presence of attacks, by adding *suitable* security mechanisms that address the detected vulnerabilities. This results in the overall framework for formal safety modeling, analysis and patching of distributed sequential automation systems under adversarial influences (Fig. 1).

Coordination between components in many IoT systems is based on discrete events. While a plethora of formal modeling frameworks is used under the umbrella of IoT (e.g., [21], [22], [23]), industrial automation systems are commonly based on GRAFCET (IEC 60848)/SFC (IEC 61131-3) control designs, and thus on the underlying formal semantics of Control Interpreted Petri Nets (CIPN). Hence, to enable existing applications to be deployed over IoT-enabled controllers, and to maintain compatibility with the domain expertise, we focus on formal security analysis of IIoT-enabled controllers described via CIPNs; such controllers may be developed directly or automatically derived from existing centralized automation designs (e.g., as in [24]).

While inherent determinism of CIPNs is not a limitation when specifying controller behaviors, it prevents the use of CIPNs to model malicious actions [15], which cannot be captured by deterministic or stochastic modeling formalisms. On

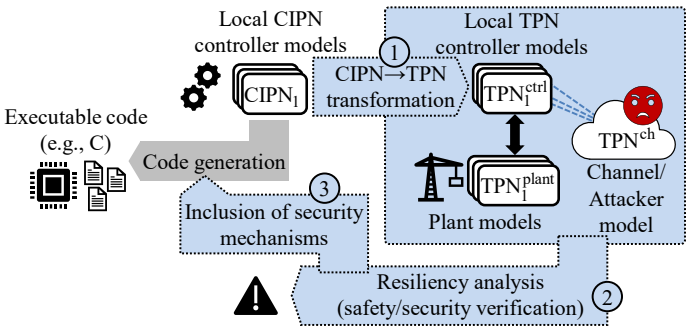

Fig. 1. Framework for resilient IIoT-based distributed automation; in Phase 1, existing distributed control models, which are used to generate executable code for IIoT controllers, are composed with channel and plant models; in Phase 2, the composition model is used to formally verify properties of interest; finally, in Phase 3, the results of the security analysis are used to enhance system resiliency, by adding suitable security mechanisms during code generation.

the other hand, Time Petri Nets (TPN) support nondeterminism, making them a great candidate for security-aware modeling. Thus, for Phase 1 of our security analysis framework, we introduce methods for automatic transformation of domain-specific CIPN-based controller specifications (i.e., designs) into TPN-compliant models. These TPN models enable closed-loop system modeling and analysis when composed with corresponding non-deterministic plant and security-aware communication channel models. We show how such security-aware models can be developed with the desired level of abstraction capturing attack impacts on automation performance. TPNs were previously used for modeling and analysis of automation systems. For example, they were successfully employed in resources optimization [25] and scheduling [26] of flexible manufacturing systems, and TPN-based A* search method was used for scheduling in robotic cellular manufacturing systems [27]. Nevertheless, this is the first work employing TPNs to capture the attack impact on control/automation systems, in a way that supports such analysis. While our framework supports any communication channel design, we focus on the IEEE 802.15.4-based implementation from our evaluation setup.

In Phase 2, on the developed security-aware closed-loop system model, we employ open verification tools (e.g., [28]) to perform system-wide verification of safety and QoC-relevant properties in the presence of attacks; note that we make no assumptions on the specific attack actions, from all possible malicious actions, nor the times when they may occur. As we show later in Sec. VI, our framework fully supports plant-state-bound safety properties. Essentially, these safety properties can capture unwanted behavior of the system occurrence of which leads to hazards, and their negation should always be satisfied. For instance, in a pick&place control scenario, the implementation should guarantee that the handled object will not be dropped in between the pick and place locations, in spite of possible attacks. This is also compatible with standards (e.g., IEC 61508) for formal functional safety verification. On the other hand, enabling security analysis within the same family of formalisms (i.e., using TPNs, closely related to the CIPN formalism used to design controllers), directly facilitates

domain-specific interpretation of the analysis results. This allows us to exploit verification results and directly address the discovered security vulnerabilities in Phase 3, by orchestrating security patches in code generation performed on the original CIPN models.

Finally, we show the applicability of our framework on a real-world industrial case study. We perform security analysis of an IIoT-enabled pneumatic manipulator system with multiple configurations such as 2-Degree-of-Freedom (DOF) pick & place and 3-DOF pick-immerse-shake-return.

Note that the developed models directly capture attacks on communication links as well as the fact that smart IIoT-based devices communicate events between each other as part of the control loop; thus, directly enabling analysis of QoC under attacks. Unlike in (e.g., gateway-based) IIoT monitoring systems, distributed IIoT-based automation is commonly not employing large multi-hop communication paths. Hence, as we show for the industrial case study, the presented analysis scales well, and does not require the use of more network-based modeling parameters, such as node centrality [29], [30].

Specifically, the contributions of this work are as follows:

- Security-aware framework for verification of system-level properties for distributed discrete-event controllers (based on CIPNs) in the presence of network-based attacks;
- TPN-based non-deterministic modeling of network-based attacks on distributed controller communication, with emphasis on capturing impacts on automation performance;
- Extension of the control software development cycle from security-aware analysis to firmware patching, to ensure correct operation in the presence of attacks by addressing the discovered security vulnerabilities that may have significant impact on automation performance under attack;
- Full-stack proof-of-concept case study based on industry-grade components demonstrating applicability of the developed secure automation framework.

This paper is organized as follows. Section II reviews related work, before the problem is defined in Section III. We introduce TPN-based security-aware modeling (Section IV) and derive the security-aware communication model (Section V). In Section VI, we present verification of relevant formal properties, as well as how verification results can be used in code generation to include security patches and improve system resiliency. Industrial case studies are discussed in Section VII, before providing concluding remarks (Section VIII).

II. Related Work

In [31], a model-based approach for simulating attacks on CPS is presented, but no formal analysis/verification is supported. In [32], additional formal security assessment of industrial CPS controllers is performed, but analysis remains constrained to high-level vulnerabilities at the level of functional models. A comprehensive formal security analysis of wireless IoT communications for a specific attack model is presented in [33]; however, QoC performance or security are considered in isolation, and evaluation of security-based implications on the underlying physical process is not supported.

Security analysis techniques for other IoT domains have recently attracted attention. For example, smart home IoT applications are formally surveyed for anomalous behavior [34], without considering formal adversarial models and implications of security vulnerabilities on system operation. Similarly, [35] introduces a dynamic policy-based enforcement system for securing against unauthorized and unwanted control scenarios, focusing only on architectures and platforms for consumer IoT applications in smart home automation. In [21], an SMT[1]-based framework for IoT security analysis is presented; yet, only abstract threat models are used, and the software architecture of IoT nodes is masked by behavioral modeling. In the domain of distributed automation, [37] presents formal attack modeling and evaluation of attacks using limited automata-based control models that do not consider realistic communication models (e.g., timing). Additional shortcoming is that attacks are not defined in the same formal semantics used for control design; thus, domain experts require additional training to utilize results of the security analysis.

Social resilience based on agents' reputation in software agents community has found use in e-commerce, e-learning, e-government, etc. [38]. Yet, it is not advisable to give benefit of the doubt to the agents and allow them to build reputation in industrial environment where the effects of attacks on IoT devices can be very severe (e.g., catastrophic damages of equipment, workers injuries) and safety issues are paramount.

Petri nets (PNs) have proven to be very useful not only for studying and modeling [39], but also for fault detection and recovery in centrally controlled discrete event systems [40]. PNs have also been used for security-aware modeling and analysis. Penetration analysis using attack trees is formalized through PNs (e.g., [41], [42]). Coordinated cyber-physical attack modeling for smart grids is done in [43], but only high-level attack scenarios are modeled, and the structure of the system components is coarsely abstracted. Modeling security risks and vulnerabilities for Unix-like software was performed (e.g., [44]) but without specifics of the underlying software architecture. Stochastic PN-based attack models are adopted for CPS threats in [45], [46], whereas [47] introduces a framework for formal reliability analysis of networked IIoT sequential control applications based on CIPNs. On the other hand, the work from [48] deals with fault detection in systems modeled by PNs. However, such fault/failure models are limited to stochastic behaviors that cannot accurately capture adversarial actions (as described in [15], [49], [50]). While cooperation and communication protocols may be modeled with PNs (e.g., [51], [52]), to the best of our knowledge, nondeterminism in PNs has not been exploited for adversarial modeling.

Consequently, in this work we address the highlighted limitations of the existing security-related techniques employed in the industrial automation domain, as well as the limiting factors preventing the use of the security-analysis methods from other IoT domains. Specifically, as described in the Introduction, we introduce a modeling and analysis framework for reasoning about performance and safety of

[1]Satisfiability Modulo Theories (SMTs) enable specification and verification of logical formulas over problems defined with predicate logic [36].

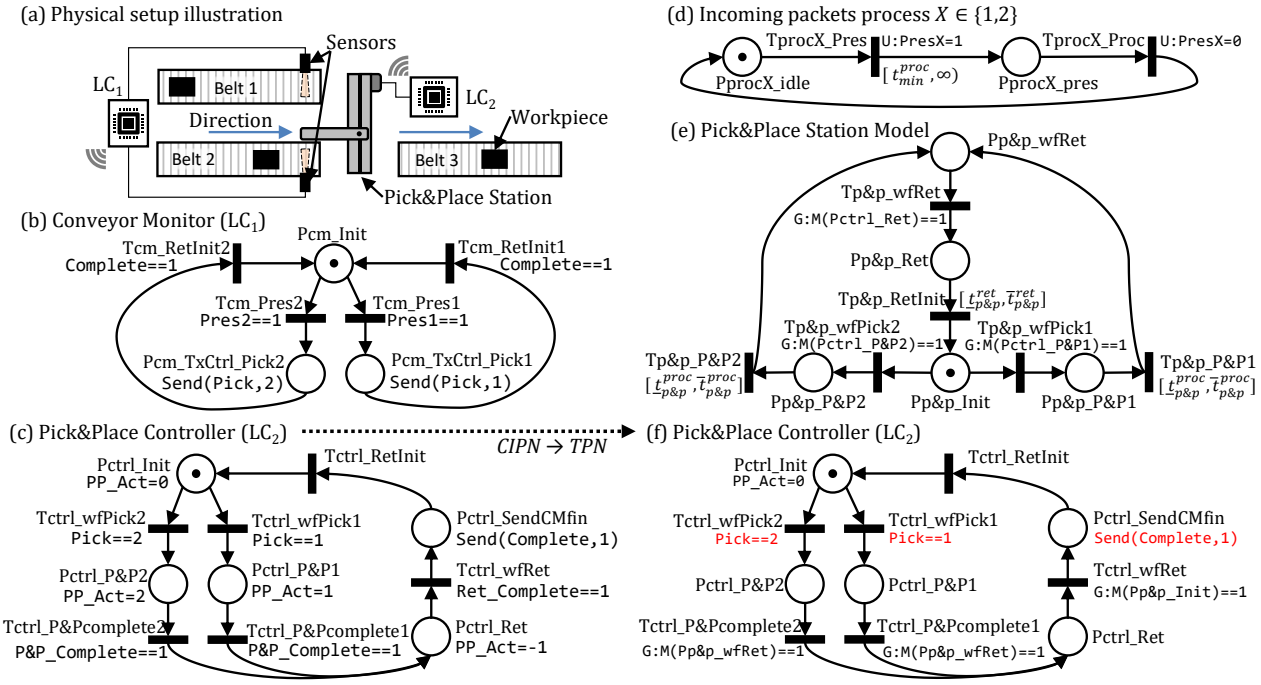

Fig. 2. Distributed conveyor/pick & place: (a) physical setup; (b) CIPN-based control model of the conveyor monitor (LC_1), (c) CIPN model of the pick & place controller (LC_2); (d) model of incoming workpieces, with a lower bound on the workpiece inter-arrival; (e) **TPN** model of the pick & place station; (f) extended model of LC_2 with TPN-compatible sensing/actuation (not a **TPN** as it still relies on the communication API (in red) for interaction with other LCs).

sequentially controlled industrial automation systems under realistic network-based attacks (caused by the use of IIoT-enabled controllers). To achieve this, our work differentiates from existing approaches in that our framework allows for:

- Holistically considering QoC performance, safety, and security of the system;
- Considering realistic communication models (e.g., incorporating timing properties of communication links);
- Modeling attacks using the same formal semantics that is employed for control design;
- Evaluating the impact of attacks on the underlying physical process and its operation.

III. Motivating Example and Problem Description

A CIPN is a 6-tuple $\mathbf{CIPN} = (P, T, F, C, A, M_0)$ where $P = \{P_1, ..., P_m\}$ is a set of places (represented by circles, see Fig. 2), $T = \{T_1, ..., T_n\}$ is a set of transitions (represented by bars) such that $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$, $F \subseteq \{P \times T\} \cup \{T \times P\}$ is the set of arcs between places and transitions, $C = \{C_1, ..., C_n\}$ is a set of logical conditions enabling synchronization of the controller with sensors by guarding the corresponding transitions in the model, and $A = \{A_1, ..., A_m\}$ is a set of actions on actuator outputs that are allocated to places.[2] The **CIPN**'s state is defined by its *marking*, i.e., the token position (captured by a dot inside the corresponding place), and transition firing (i.e., token flow) represents a state change; M_0 is the initial marking. CIPN semantics is deterministic [53] (for formal semantics see [54]). For distributed automation, functionality of each local controller (LC) is captured by the corresponding CIPN. For event data exchange between LCs, places of a CIPN may invoke the communication API exposed by the LC runtime environment; this is denoted as `Send(signal,value)` for broadcast or `Send({dest1,dest2,...},signal,value)`[3] for uni/multicast transmissions. Dually, the receiving LC can condition its transitions with statements similar to conditioning on locally connected sensors (i.e., as `signal==value`) [24].

Example 1: Consider a simple control system from Fig. 2(a) with the conveyor monitor (LC_1) and the pick & place station controller (LC_2). Two LC_1 sensors for the parallel incoming conveyor belts sense if a workpiece is ready to be picked from either of the conveyors and placed on the third, outgoing conveyor. Fig. 2(b-c) show CIPN-basd controllers for LC_1 and LC_2. Initially, LC_1 is in state `Pcm_Init`, waiting for either of its sensors $X \in 1, 2$ to indicate workpiece presence (i.e., transition `Tcm_PresX` is conditioned by the sensing event `PresX==1`).[4] Upon detection of a workpiece, LC_1 sends a message to LC_2 (via API call `Send(Pick,X)` in place `Pcm_TxCtrl_PickX`) indicating the conveyor with a workpiece to be picked. LC_1 then waits for completion of the pick & place. The rest of the workcycle similarly follows.

These CIPN control models assume ideal communication links, without unpredictable channel behaviors. For instance, consider an adversary with network access who mounts an

[2]Logical conditions dependent on system sensors are denoted as `sensor==value`, while actuation commands associated with places in the form `actuator=command`.

[3] In this work, we use different font (i.e., as `Send`) for CIPN and TPN model primitives (e.g., code API, names of variables, places and transitions).

[4]For model readability, we employ descriptive notation for places, transitions, conditions, and actions; e.g., transition `Tctrl_wfRet` in controller LC_2 waits for the return cycle to finish, while place `Pcm_TxCtrl_Pick1` on the conveyor monitor sends signal `Pick==1` to the pick & place controller.

impersonation (i.e., spoofing or masquerade [18]) attack while LC_2 is waiting for a message from LC_1 that a workpiece should be picked up. This can be performed by exploiting the lack of authentication services, enabling the attacker to transmit messages on behalf of any of the controllers. For instance, by sending the corresponding message (e.g., by signaling `Send(Pick,1)`), the attack will result in the pick & place station pickup by LC_2; hence, it may collide with upcoming workpieces, potentially incurring mechanical damage, or a wasted workcycle. Similar holds for *message modification* [18] (i.e., signal replacement [55]) attacks, when the right conveyor belt contains a workpiece ready to be picked up (i.e., `Pres_R==1`), but the attacker intercepts the corresponding message and maliciously signals `Send(Pick,2)`. With a wireless channel between the controllers, this may be achieved by jamming original transmissions until the legitimate controller's retries are exhausted, and then transmitting the malicious signal. Also, if an adversary *delays* or *blocks* some transmissions or acknowledgements (ACKs) between LCs (i.e., by launching a Denial-of-Service (DoS) attack [18]) the system may experience excessive downtime (e.g., due to an attack that is maliciously occupying the shared channel). □

These examples illustrate that distributed control may be affected by an attacker with network access. Hence, we focus on security aspects of IIoT-enabled distributed automation systems; our goal is to provide methods to model and analyze system behaviors in the presence of *network-based attacks*, while enabling the use of analysis results to modify (i.e., update) the system in order to achieve attack-resilient operation.

Overview of our Approach

We start from a functional description of N LCs expressed as $\mathbf{CIPN}_i$, $i = 1, ..., N$. We consider an attacker with full access to the network with M communication channels. The attacker is not able to compromise the LCs, but has full knowledge of the state of each LC. Our design-time framework illustrated in Fig. 1 starts from automatic transformation of CIPN control models to TPN-based models; such models enable explicit capturing of the (i) communication semantics, (ii) platform-based effects using timed transitions to model non-zero execution and message propagation times, and most importantly (iii) the non-deterministic behaviors necessary to model adversarial actions.

We show how the remaining closed-loop system components (i.e., the plant and communication channel in the presence of attacks) can be modeled using the TPN formalism. Furthermore, we demonstrate how composition of these models enables system-wide analysis of control performance in the presence of attacks. Finally, we show how design-time formal verification results can be used during code generation for smart IIoT-based controllers, which facilitates updates of LCs' firmware to address exposed security problems.

Remark 1 (Petri nets vs Automata/Finite-State Machines): We build on Petri Net-based modeling since CIPNs are the main formalism used to capture existing (including distributed) automation systems. For example, GRAFCET (IEC 60848) and SFC (IEC 61131-3) languages for programming industrial control systems originate from Petri Nets, with their behavioral equivalence discussed in [54], [56], [57]. However, the proposed framework, including network and attack modeling, can be directly extended to other discrete-event IoT systems whose behavior can be captured with automata/finite-state machines, since formal mappings between semantics of Petri nets and automata have been defined (e.g., as in [58], [59]). □

IV. TPN-Based Automation Modeling

TPNs extend CIPNs by introducing timed transitions. Formally, TPN is a 6-tuple $\mathbf{TPN} = (P, T, F, V, D, M_0)$ where (P, T, F, V, M_0) is the corresponding Petri net and D is the duration function defining the transition firing times in an interval $(\underline{t}_f, \overline{t}_f), [\underline{t}_f, \overline{t}_f), (\underline{t}_f, \overline{t}_f]$, or $[\underline{t}_f, \overline{t}_f]$.[5] Here, $\underline{t}_f$ and $\overline{t}_f$ are the lower/upper bound on the transition firing times, which may also be *zero* or *infinity*, whereas time interval next to *immediate* transitions (i.e., with zero firing time) is not specified. Firing times are defined relative to the moment of transition enabling, without any assumptions on their distribution. Therefore, non-determinism (i.e., non-deterministic behaviors) is introduced in TPNs, enabling modeling of timed properties of real-time control software [61], [62], [63]. Additionally, modern TPN analysis tools (e.g., [28]) support definition of guard and update functions in the same way as in CIPNs (described in Sec. III) and thus facilitate modelling of attacks through straightforward addition of CIPNs already developed during control system design.

Therefore, we transform formal distributed control specifications expressed by $\mathbf{CIPN}_i$, $i = 1, ..., N$, into TPN-compatible models $\mathbf{TPN}_i^{ctrl}$, $i = 1, ..., N$. We then compose these models with plant models $\mathbf{TPN}_i^{plant}$, $i = 1, ..., N$, and security-aware communication channel models $\mathbf{TPN}_j^{channel}$, $j = 1, ..., M$,[6] which enables us to reason about system-level safety and security properties under the modeled adversarial influences. Since both CIPNs and TPNs originate from PNs, the translation from $\mathbf{CIPN}_i$, $i = 1, ..., N$ controller models to $\mathbf{TPN}_i^{ctrl}$, $i = 1, ..., N$ is direct for all places and transitions except where platform-implemented API is called, i.e.,

1) Places handling actuation, and transitions handling sensing by issuing I/O API calls (`actuator=value` and `sensor==value`, respectively),
2) Places handling transmissions and transitions handling receiving of communication signals via API calls (`Send(destination,signal)` and `signal==value`, respectively),
3) Places calling other platform-dependent API, such as request for execution delays.

These CIPN constructs, which directly rely on the underlying platform used to implement the controller, must be explicitly modeled as nets that capture: 1) interaction between LCs $\mathbf{TPN}_i^{ctrl}$ and the plant $\mathbf{TPN}_i^{plant}$, 2) interaction between LCs $\mathbf{TPN}_i^{ctrl}$ and communication channel(s) $\mathbf{TPN}_j^{channel}$, and

[5] Formal semantics of TPNs is described in detail in, e.g., [60].

[6] This captures the general case where time or frequency multiplexing may be used to provide multiple communication channels over the same medium.

3) runtime environment changes based on issued commands (e.g., variable updates, execution delays).

Thus, we first introduce methods for automatic extraction of TPN-based controller models from existing CIPN models. We then capture interaction between the automation system and plant, as well as the LC platforms' runtime environment (all in Section IV-A); this is followed by security-aware modeling of communication channels and their interfaces with LCs (Section V). These methods produce a full closed-loop system model that enables reasoning about system resiliency to attack.

A. Modeling Plants and Controller-Plant Interaction

Nominal behavior of the physical plant is typically known at control design time. Since the CIPN formalism is universally adopted for automation design, we thus assume that a PN-based (i.e., **CIPN** or **TPN**) plant model is available.[7] On the running example, we describe development of such TPN plant model, along with a TPN-compliant controller-plant interface implemented through marking-dependent *guard functions*.

Plant Modeling: Fig. 2(d) models the incoming workpieces arrivals with a lower bound on the interarrival times, whereas Fig. 2(e) shows a **TPN** model of the pick & place station from Fig. 2(a). Place `Pp&p_Init` represents the station's initial state. Token flow from this place is conditioned by the corresponding commands of LC_2 `PP_Act==1` and `PP_Act==2` from the LC_2 model in Fig. 2(c). In the TPN formalism, *marking-dependent guard functions* can be used to restrict state changes (i.e., token flow); namely, a marking-dependent function, denoted by `M(·)`, assesses the state of the controller model (i.e., token distribution) and returns the current number of tokens inside the argument place. Hence, guard function `G:M(Pctrl_P&P1)==1` (or `G:M(Pctrl_P&P2)==1` for the 2nd conveyer belt) is associated with transition `Tp&p_wfPick1` (or `Tp&p_wfPick2`).

For example, once the pick & place process is triggered by the LC_2's model advancing its token to `Pctrl_P&P1` (or `Pctrl_P&P2`), the station's token transitions to place `Pp&p_P&P1` (or `Pp&p_P&P2`) if conveyor belt 1 (or 2) has a workpiece waiting to be processed. Note that to capture realistic executions, the actual times to complete the pick & place and return processes are not deterministic; transitions `Tp&p_P&P1` / `Tp&p_P&P2` have firing times from the interval $[\underline{t}^{proc}_{p\&p}, \overline{t}^{proc}_{p\&p}]$, as shown in Fig. 2(e).

Modeling Control-Plant Interaction: The actuation part of the plant-controller interface is managed by guard functions assessing the controller's marking; thus, explicit actuation input updates (e.g., `PP_Act=1`) in CIPN places are omitted in the transformation to the TPN model, because TPN places do not feature any attributes. This interface can also be achieved with *update functions* that are triggered on the firing of controller's transitions to update markings or variables. The choice of the transformation semantics from CIPN to TPN can therefore be adjusted to the specific platform implementation.

[7]On the other hand, if an automata or other discrete-event system representation of the plant is available, existing tools and methodologies can be used to translate such models into a PN-based representation (e.g., [58], [59]).

Similarly, sensing is modeled by introducing plant-marking-dependent guard functions on controller's transitions. Specifically, transitions conditioned by sensor values in the form `sensor==value` in a CIPN model are replaced with immediate transitions guarded by a Boolean function evaluating to *true* if the plant model marking corresponds to the plant state where `sensor==value` is satisfied, and to *false* otherwise.

For instance, once LC_2 commands return of the pick & place station, (i.e., LC_2 model from Fig. 2(c) has the token in `Pctrl_Ret`), it is blocked on the transition `Tctrl_wfRet` guarded by condition `Ret_Complete==1`. This transition in the CIPN model is transformed into a transition in the TPN model in Fig. 2(f), guarded by a function dependent on the marking of the pick & place station model from Fig. 2(e) (i.e., controller waits for the station to reach home position). Guard function `G:M(Pp&p_Init)==1` returns *true* once the token in the plant model transitions to `Pp&p_Init`; hence, LC_2 can transition over `Tctrl_wfRet`. Therefore, this guard function is used for the transition `Tctrl_wfRet` in the TPN model of LC_2. More complex conditions based on multiple sensors are implemented by forming an arbitrary plant marking-dependent Boolean guard function.

Finally, Fig. 2(f) shows a controller model of the described controller-plant interface. The model, obtained from the CIPN-based model in Fig. 2(c), is intermediary, and not fully TPN-compliant; the CIPN-based communication semantics (i.e., signal transmissions via `Send(destination,signal)`, and receptions through `signal==value` denoted in red in Fig. 2(f)) is still present in the model. However, to allow for verification of system properties when networking is not a concern, this communication semantics can be easily adapted to TPNs by applying the same *guard/update* functions as described; this results in a model architecture from Fig. 3(a).

1) Controller Runtime Environment Modeling: Another challenge for the automatic mapping of CIPN-based control models into TPN-compliant models is mapping of places issuing system calls from the runtime environment (e.g., execution delays, requests for timer interrupts, setting counter events) or updating local controller state (e.g., manipulating global variables). Requested execution delays can easily be modeled as timed transitions with the exact firing times (i.e., where the lower and upper firing time bound are the same); in general, however, event timings with different semantics are available depending on the control implementation—i.e., GRAFCET or SFC. In [56], [57], authors provide detailed translational semantics between CIPNs and TPNs in these cases by introducing *event sequencers* as certain conditions exist where transitions can be taken while time to some events generated internally in places has still not elapsed.

Remark 2 (Modeling more Complex Execution Environments): While we consider single-threaded automation examples (as most sequential control implementations are), existing techniques for modeling parallel systems can be applied given the expressiveness of TPNs. For example, for multithreaded applications where task preemption is allowed, the operating system scheduler can be modeled as a separate component, even in case of multi-processor platforms [64]. □

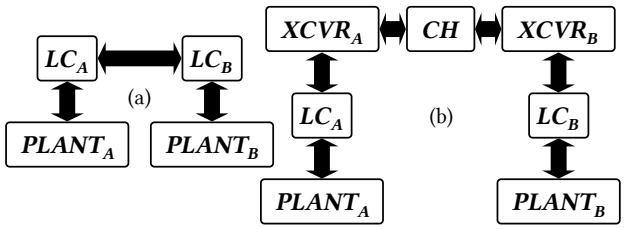

Fig. 3. Model architecture: (a) Model captures local controllers LC_i, plants $PLANT_i$ and their interactions, (b) Model also captures the employed communication transceivers $XCVR_i$, and the underlying communication channel CH_i. Note that local controller models LC_i in schemes (a) and (b) are not the same; i.e., in (b), controller places/transitions invoking communication APIs are made compatible with the transceiver model.

B. CIPN and TPN Controller Equivalence

An execution path in CIPNs can be defined as a sequence of markings, where a change in the marking occurs due to firing of a transition. Recall that places are associated with actions; hence, each marking is associated with a set of actions, while transitions are associated with guards—firing of each transition is thus conditioned by a set of conditions.[8] Therefore, an execution path is a sequence $M_0, \mathcal{T}_1, M_1, \mathcal{T}_2, ...$, where $\mathcal{T}_i$ is the transition taking the net from marking M_i to M_{i+1}. In the TPN model, a path is characterized by a similar sequence with the addition of transition timing.[9] In our case, it is sufficient to maintain the CIPN *controller* execution paths in the TPN model, as our objective is operational equivalence of the source CIPN, and the target TPN control models.

CIPN controller specifications are fully deterministic by design, and have only *immediate* transitions.[10] Thus, the target TPN controller models automatically obtained without additional constructs (i.e., as described in Sec. IV-A), do not introduce behaviors not covered by the source CIPN models. Consequently, execution paths of the composition of the TPN models match that of the CIPN, from the input-output (i.e., sensing-actuation) perspective. In other words, no execution path is added by transforming the CIPN controller into the TPN representation [24]. Intuitively, the TPN models obtained by direct mapping from CIPN (i.e., place-by-place, and transition-by-transition) are still *fully deterministic* (isolated from the intrinsically non-deterministic plant and channel); i.e., their behavior is identical to their CIPN counterparts, and same behavioral assumptions (e.g., 1-boundedness) hold [54].

V. Security-Aware Modeling of the Channel and Controller-Channel Interaction

We now introduce a security-aware channel model, including a TPN-compliant controller-channel interface that enables model composition. Hence, we address modeling challenges to enable the transition from the security-agnostic model structure from Fig. 3(a), to the security-aware model composition shown in Fig. 3(b). We start by defining the attack model.

A. Attack Model

We assume a powerful network-based adversary that has:

1) The full knowledge of the system, including the CIPN models, generated code and analysis framework, as well as the current state of all LCs (and their transceivers);
2) Network access and full communication protocol compliance, i.e.,the attacker is able to transmit unsigned messages as any of the LCs, or intercept messages or ACKs exchanged by LCs;
3) The ablility to precisely time actions and align transmissions with legitimate network traffic, e.g., to interfere with legitimate messages by transmitting the carrier signal or a protocol-compliant message.

Therefore, the attacker may mount the following attacks:

1) *Interception or delaying of legitimate packets (DoS):* With these attacks, adversarial transmissions occupy the communication channel, (a) blocking transmissions from legitimate LCs to prevent or delay their access to the network, or (b) blocking ACKs on legitimate LC's transmissions to cause unnecessary retransmissions and slow down progression of the targeted transmitter [18];
2) *ACK spoofing:* The attacker may impersonate an ACK expected by a legitimate transmitter; e.g., following by interception of the transmission, the attacker may spoof the ACK misleading the legitimate transmitter into believing that the transmitted signal was received by the intended receiver [17], both for regular and 'heartbeat'/sync messages [65].
3) *Impersonation/Masquerade:* The adversary may transmit false event signals on behalf of a legitimate LC (i.e., impersonating another controller), with the goal to inject false commands [17] or sensor measurements; such attacks could e.g., allow the targeted receiver to resume execution while it is blocked waiting for an event signal, before the event it is sent by the legitimate LC.
4) *Signal replacing/Message modification:* The adversary may modify content of a legitimate message to deliver false event information. While logically the same, the attack procedure differs from intercepting a legitimate transmission followed by a masquerading attack [19], and thus is modeled differently.[11]
5) *Replay attack:* The attack characterizes an adversary that records events signaled by the LCs and replays the sequence of events on behalf of one or more LCs; thus, maliciously emulating activity of LCs whose operation (s)he is interfering with [19].

This attack set covers all reported attacks that, from the standpoint of low-level signaling of events, could have direct impact on QoC of the underlying physical process [15], [66]. Other attacks, such as attacks on network routing policies,

[8]We employ the standard assumption that all inputs are re-evaluated after firing of every transition (e.g., as done in [57]).

[9]Strictly, two types of time intervals characterize each transition: static intervals (i.e., design-time bounds) when they may fire, and dynamic (i.e., runtime) intervals when they can fire at any given instant, conditioned by all other enabled transitions. However, for purposes of showing marking-based equivalence with CIPNs, time can be abstracted away.

[10]Immediate transitions are fireable imediately after enabling, without delay.

[11]This type of attack is technically more challenging to perform compared to other attacks, especially over a wireless medium.

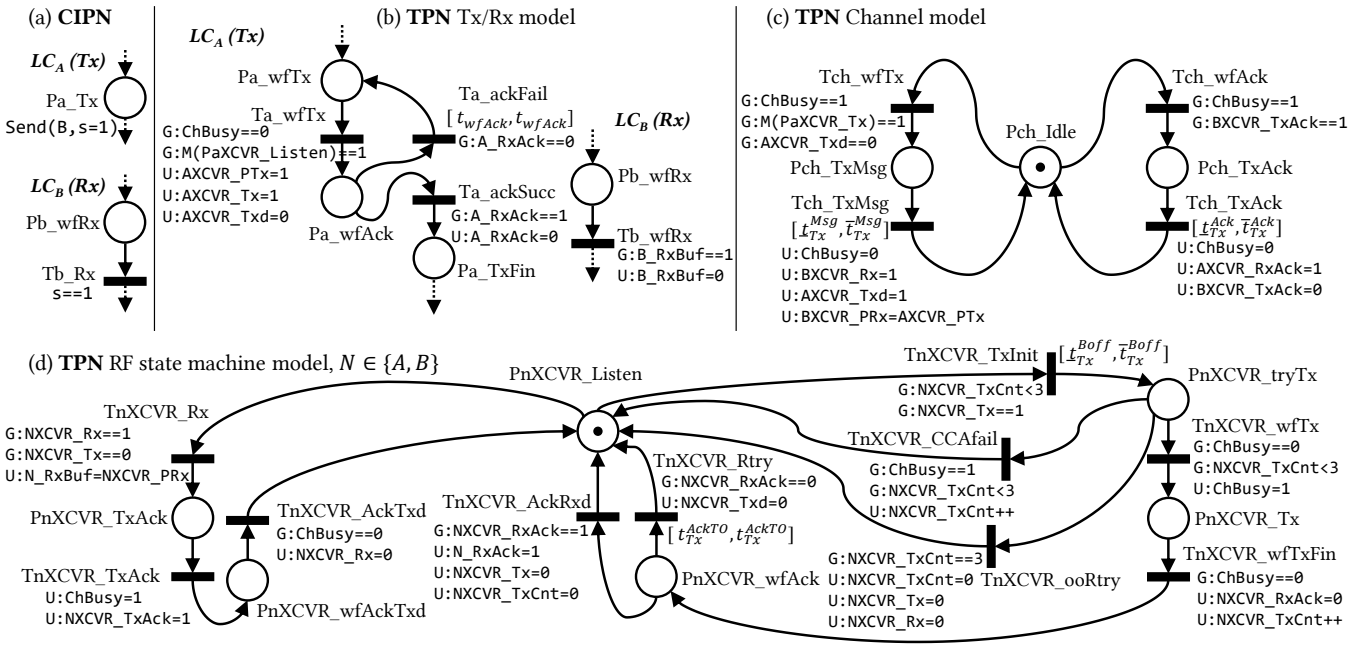

Fig. 4. Transformation between CIPN-based and TPN-compatible communication models; (a) a Tx/Rx place/transition pair in the CIPN formalism; (b) the same Tx/Rx place/transition pair modeled with as a TPN adjusted to the half-duplex, acknowledge-required unicast CSMA-CA-based channel, whose model is shown in (c); (d) model of the employed radio transceiver (i.e., the governing RF state machine TPN model). Note that each Tx/Rx net pair in (a) (from the model in Fig. 3(a)) is extended into a corresponding pair in (b), while only a single model from (c) and (d) are added to obtain the model from Fig. 3(b).

are focused on higher-level information flows and are thus harder to directly relate to the automation QoC [19]. Accordingly, our goal is to model attacks by capturing their influence on the sequential control system and the resulting QoC, rather than the employed attack vector for any specific attacks; i.e., the attack model should be agnostic to the actual attack implementation. By abstracting away attacker's actions in the physical sense, we achieve the balance between model expressiveness and complexity without any loss of information from the perspective of the control system under attack.

B. TPN-Based Modeling of Attack Impact

Recall that CIPN models rely on platform-provided communication APIs for passing events between LCs; e.g., as in Fig. 4(a), `Send(destination,signal=value)` command within a place sends the updated `value` of `signal` to the `destination` LC, while condition `signal==value` on a transition within the model blocks execution until the signal corresponding to the desired value is received over the network. To enable formal analysis of the attack impact on QoC of distributed automation, it is necessary to develop a TPN-compliant model of the interface (i.e., transceiver) between the controller and security-aware channel model; such model can be then composed with the TPN-based models described in Section IV-A, resulting in Fig. 3(b) architecture.

Such security-aware formal model has to capture: (1) application-level (i.e., controller side) communication stack behavior, directly affected by (2) the channel-side (i.e., communication medium) attack model, and (3) the controller-channel interface. Specifically, application-level (i.e., control-related) communication stack behavior, such as delays or blocking on communication peripheral resources, is of interest for security analysis, as this presents the main reflection of the communication-level attacks onto the control functionality.

Therefore, when translating the CIPN communication model from Fig. 4(a) into a TPN-compliant model, it is necessary to capture application software states that directly affect progress of the control functionality, conditioned by data dependencies resolved via communication. Such models can be obtained from the actual application firmware running on the embedded LCs (i.e., source code). For example, when IEEE 802.15.4 protocol is used, as in the case study presented in Section VII, the state-machine/TPN representation can be directly extracted from the radio driver (as done in Fig. 4(b)). On the other hand, if a more complex communication stack is considered (i.e., also implementing higher networking layers), exiting state-machine extraction techniques (e.g., [34]) can be used.

Second, the channel model has to explicitly capture the channel states essential for supporting the attack models presented in Section V-A; channel features that are not observable (or alterable) need not be modeled (e.g., bit-level signaling, or carrier-level modulation). Finally, a TPN-compliant interface between the controller and security-aware channel models is needed to allow for their formal composition, enabling system-level analysis of adversarial influence on the entire system. Therein, specific data link layer (OSI model Layer 2) features are crucial for understanding retransmissions and ACK mechanics which, as we will show, affects design of attack detectors. Therefore, while controller models should capture application-level communication semantics, it is also necessary to include protocol-level details within the transceiver (XCVR) models, which act as the interface between the controllers and the communication medium (Fig. 3(b)). XCVR specifics are commonly available for the specific employed radio commu-

TABLE I
SYMBOLS USED IN FIG. 4 AND FIG. 5; THIRD COLUMN (WHERE APPLICABLE) INDICATES ACCESSIBILITY TO APPLICATION SOFTWARE (OR ONLY TO THE TRANSCEIVER'S INTERNAL RF STATE MACHINE).

Symbol	Description	SW acc.
`ChBusy`	Indicator whether the channel is currently busy with a packet or ACK	YES
`N_RxBuf`	Local Rx buffer	YES
`N_RxAck`	Local flag indicating successful transmission, i.e., ACK reception	YES
`NXCVR_PTx`	Transceiver Tx payload buffer	YES
`NXCVR_PRx`	Transceiver Rx payload buffer	YES
`NXCVR_Tx`	Signal to XCVR initiating transmission	YES
`NXCVR_Txd`	Signal to XCVR indicating transmission	NO
`NXCVR_Rx`	Signal from XCVR indicating reception	YES
`NXCVR_TxAck`	XCVR signal initiating ACK transmission	NO
`NXCVR_RxAck`	XCVR signal indicating ACK reception	NO
`NXCVR_TxCnt`	XCVR retry counter	NO
$\underline{t}_{Tx}^{Msg}, \bar{t}_{Tx}^{Msg}$	Message transmission time (bounds)	—
$\underline{t}_{Tx}^{Ack}, \bar{t}_{Tx}^{Ack}$	ACK transmission time (bounds)	—
$\underline{t}_{Tx}^{Boff}, \bar{t}_{Tx}^{Boff}$	Back-off time (bounds)	—
t_{Tx}^{AckTO}	Data link layer ACK timeout	—
t_{wfAck}	Application-level ACK timeout	—
$\underline{t}_{Ch}^{DoS}, \bar{t}_{Ch}^{DoS}$	Contention time due to DoS (bounds)	—

nication chip as RF circuitry control is usually state-machine based (e.g., referred to as the *internal RF state machine* [67] in the case of radios used in our implementation).

Finally, explicit security-aware channel modeling is *medium-, protocol-, and attack-dependent*. Fig. 4(c) and Fig. 5 show a security-aware model of a *half-duplex, acknowledge-required unicast CSMA-CA-based communication channel* with respect to the previously defined attack model. While other medium/protocol variants can be easily modeled due to the expressiveness of TPNs, we consider this model as it applies to our physical setup described in Section VII. In the rest of this section, we describe the transformation from the CIPN-based LC communication model to a TPN model assuming the aforementioned channel, while aiming to balance between the model expressiveness and capturing security-aware behavior required for analysis of QoC under attack.

Notice that in this model, each and every place modeling a transmission (as in Fig. 4(a)) translates to three places and three transitions (one of which is timed) with additional guard and update functions; furthermore, the receiving place-transition pair remains the same with the addition of one update function. Since only one transceiver model (Fig. 4(d)) is required per controller, and only one channel model (Fig. 4(c)) is required for the entire control model, the model grows only linearly in complexity during the described transformation.

C. Security-Aware Modeling of the Channel and Controller-Channel Interaction

Fig. 4(b) shows the TPN transmitter/receiver models that replace the platform-independent CIPN transmitter/receiver model in Fig. 4(a). Fig. 4(c) shows the nominal channel model (i.e., without adversarial influences), while Fig. 4(d) shows the transceiver (XCVR) model. Notice that both LC_A and LC_B have identical transceivers; thus, $N \in \{A, B\}$ in place/transition names. Table I enumerates symbols (local flags, variables, and transition timing parameters) used in the models in Fig. 4, 5. The internal RF state machine can be in the *listening*, *transmitting a packet*, *waiting for acknowledgement*, or *transmitting an acknowledgement* states. The transceiver employed in our case study (in Sec. VII), performs up to three retransmissions before signaling a transmission failure to the application. On the application level, an unbounded number of retransmissions are performed in case the transceiver returns failure. The TPN model in Fig. 4(b-d) models this interaction.

In the remaining of this section, we show how the attacks described in Section V can be modeled as TPNs. Specifically, we describe additional places, transition, and arcs to be added to the nominal channel model shown in Fig. 4(c) to capture the attacks. To enhance model readability, Fig. 5 depicts only additional places and transitions in red color required to model a specific attack, while the nominal places and transitions are depicted in black (all parts of the nominal model not relevant for the specific attack are omitted therein).

a) DoS attack submodel: Fig. 5(a) shows the DoS attack submodel. When the channel is idle, the attacker may decide to occupy the channel to prevent legitimate transmissions. He/she may do so at any time (non-deterministic choice) when the channel is not busy, and keep the channel busy arbitrarily long. In the model, the channel is kept busy for some non-deterministic time in the range $[\underline{t}_{Ch}^{DoS}, \bar{t}_{Ch}^{DoS}]$, after which it is released by the attacker.

b) ACK interception/spoofing submodel: Fig. 5(b) shows the ACK intercept/spoof submodel. To model ACK interception, an additional transition is needed allowing the channel to return to idle state following ACK transmission, without the transmitter (LC_A) receiving the ACK sent by the receiver (LC_B); i.e., transition `Tch_TxAckInt` is added as shown in Fig. 5(b), and is *not* associated with the update function `U:BXCVR_RxAck=1`. However, when ACK spoofing is considered, the attacker may transmit an ACK when the targeted receiver *is not* in the process of acknowledging, while the targeted transmitter *is* in the process of waiting an ACK.

Additionally, malicious ACK spoofing may be performed when the signal to which the ACK is intended to correspond is transmitted already, but the ACK has not yet been received by the sender (e.g., due to an intercepted ACK from the legitimate receiver). This is enabled with the additional net branch in Fig. 5(b) starting with transition `Tch_wfAckImp`. As a result of firing of this transition, the channel is declared busy and the spoofed ACK is assumed to take the same time as transmitting legitimate ACKs; thus, the transition `Tch_TxAckImp` has the same attributes as `Tch_TxAck`, with the exception of the signal to the targeted transmitter signalling ACK transmission is done (i.e., update `U:AXCVR_RxAck=1` is omitted).

c) Message intercept/modify submodel: Fig. 5(c) shows the message intercept/modify submodel, where an additional transition `Tch_TxMsgMod` (`Tch_TxMsgInt`), represented as one transition for conciseness, is added. In the case of the modification attack, this transition in the model allows the attacker to deliver a signal different from the one originally transmitted (i.e., `U:BXCVR_PRx=Pmod` where `Pmod` is the payload modified by the attacker). In the case of packet

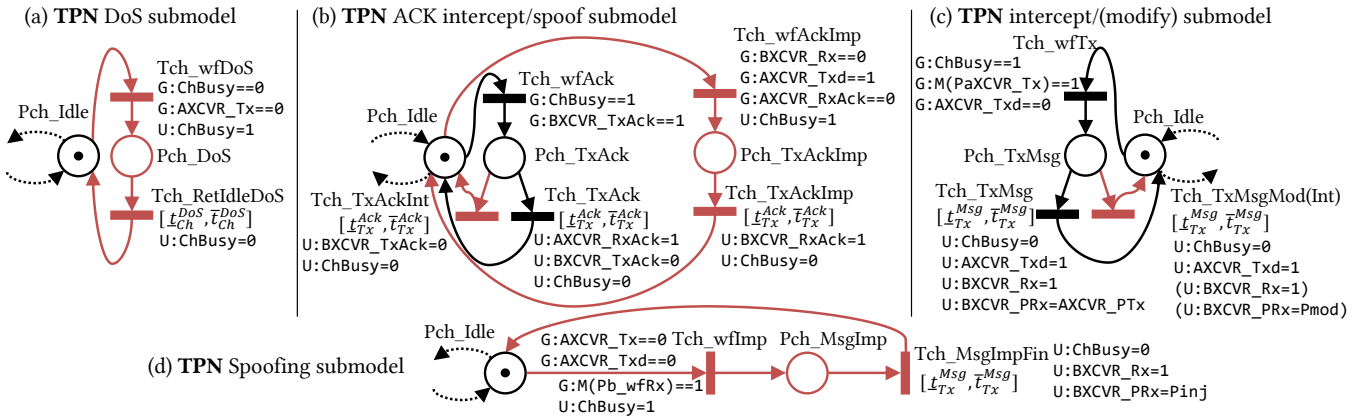

Fig. 5. Additional places, transitions, and arcs required to obtain a security-aware channel model for different attack types: (a) DoS, (b) ACK intercept/spoof, (c) message modification, and (d) masquerade. All attack-related components of the model are depicted in red color, while nominal components are shown partially for completeness in black (where relevant).

interception, no update to the receiver's XCVR buffer is made, and consequently the XCVR is not notified of a received packet (i.e., update functions are omitted and denoted as `(.)` in Fig. 5(c)).

d) Message impersonation submodel: Fig. 5(d) presents the masquerade submodel. The additional transitions and places allow the attacker to make a non-deterministic choice to impersonate transmission of the expected transmitter whenever the channel is not busy, the targeted receiver is waiting for the corresponding signal, and the original transmitter is not in the process of sending this signal. Then, similarly to the nominal (legitimate) transmission model (Fig. 4(c)), the transmission takes a non-deterministic time in the same range as legitimate transmissions. The received payload on LC_B is in this case the value `Pinj` crafted by the attacker, rather than `AXCVR_PTx`, normally transmitted by LC_A in the adversary-free case.

Remark 3 (Replay attacks): Due to the introduced non-determinism, any specific sequence of attack actions are contained within the presented model (as long as the individual actions correspond to the attack model from Section V). Thus, replay attacks are covered by the presented model as they are only specific executions of the presented security-aware channel model. On the other hand, using a similar approach, finite memory replay attacks can be captured by a model that restricts inserted attack signals only to the previously transmitted messages, as done in [15]. □

Remark 4 (Controller-plant VS controller-channel interface modeling fidelity): Control interface to the channel is modeled in far more detail than the interface to the plant, by abstracting away locally-connected actuator drives, relays, analog amplifiers, etc. The reason is that, in this work, we do not consider physical plant-level attacks. Hence, modeling the controller-plant interaction at a lower level of abstraction would unnecessarily increase model complexity. However, the presented techniques can be easily extended and the framework fully adapted to also cover physical attacks on the plant. □

VI. Resiliency Analysis and Security Patching

A security-aware closed-loop system model obtained by composing the developed security-aware TPN models can be used to verify system-level safety and QoC properties in the presence of attacks. TPN analysis tools (e.g., [28], [68]) allow for verification of formal properties specified as Linear Temporal Logic (LTL), Computational Tree Logic (CTL), or Timed CTL (TCTL) formulas [69]. In this work, we employ the tool Romeo [28] that enables verification of TCTL-based formal queries, such as traditional safety (e.g., 1-boundedness [54]) and liveness properties (e.g., absence of deadlock). In addition, as plant models are included, we can specify relevant domain-related plant-state-bound properties that are crucial for functional safety and QoC assessment. For our running example, the considered properties include:

Property 1: A workpiece on conveyor 1 never triggers a pick-up from conveyor 2; this can be formally captured as: `AG(not(M(Pp&p_P&P2)==1 and activeConveyor==1))`,[12] where `A` and `G` are quantifiers signifying formula satisfaction along *all* paths and *always* (i.e., along all subsequent paths), respectively.

Property 2: A workpiece detected on any of the conveyors is eventually picked-up: `(M(Pcm_TxCtrl_Pick1)==1 or M(Pcm_TxCtrl_Pick2)==1)-->(M(Pp&p_wfRet)==1)`, where `-->` denotes the "leads to" property; i.e., `p-->q` means that for all executions, continuous satisfaction of property `p` implies always eventual satisfaction of property `q`, or formally `AG(p => AF(q))`.

Property 3: The pick & place station does not commence cycle (i.e., it is neither in `Pp&p_P&P1` nor `Pp&p_P&P2`), while the conveyor monitor is waiting for incoming workpieces (i.e., in the place `Pcm_Init`). Formally, `AG(M(Pcm_Init)+M(Pp&p_P&P1)+M(Pp&p_P&P2)<=1)`.

Using the Romeo tool, we verified that these as well as other QoC- and safety-critical properties are *not* satisfied in the presence of attacks, as the attacker is capable of significantly altering the intended interaction between LCs, at arbitrary moments in time. For example, Property 1 is violated under *message modification* attacks, Property 2 under possible infinite *DoS*, while Property 3 fails under *spoofing*.

[12]Variable `activeConveyor` is set when a workpiece presence is detected (i.e., on transitions `Tcm_Pres1` or `Tcm_Pres2` of the conveyor monitor, shown as CIPN in Fig. 2) and reset when conveyor monitor returns to initial state (i.e., over `Tcm_RetInit`)

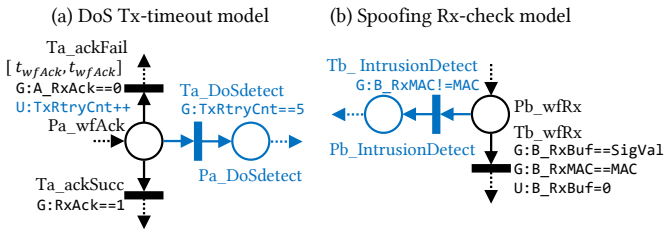

Fig. 6. Model adaptation to addition of security services; (a) for DoS detection, and (b) against spoofing. Additional places and transitions are shown in blue color.

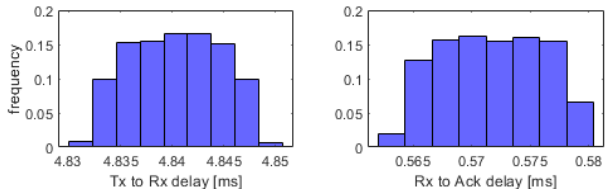

Fig. 7. Tx-to-Rx and Rx-to-Ack times measured on our IEEE 802.15.4-enabled LC platform described in Section VI.

The aforementioned properties are violated regardless of the values of timing parameters used in the model. Note that bounds on time-to-transmit and time-to-acknowledge can be obtained from experimental measurements, or directly from network specifications. Also, transceiver-related timings (e.g., back-off time during clear channel assessment) can be obtained from the employed transceivers' specifications.

Regarding verification scalability—in the system model, one nominal pick & place cycle, with all attacks disabled, contains around 35 transitions, which is on the order of the number of states in the model. Model complexity increases with the addition of non-deterministic attack choices, besides the time-induced non-determinism (in the plant model).[13] Yet, in all cases, the tool takes less than 1 s to find an execution path violating the properties, on a workstation with an Intel i7-8086K CPU (4 GHz clock) and 64 GB memory.

A. Addressing the Discovered Vulnerabilities

As our previous analysis have shown, attack actions may significantly affect performance of distributed IoT-based industrial automation systems; to address them, it is necessary to add certain security mechanisms. In this section, we discuss how such security mechanisms affect system models and verifiability of the relevant properties.

1) Detecting DoS Attacks: Packet and acknowledgement dropouts are common in wireless communication, and hence ACK and retransmission mechanisms are commonly used in such setups. For instance, in our experimental setup described in Sec. VII, ACK request can be disabled in transceiver settings, in which case no retransmissions are attempted on the data link layer. For two isolated transceivers, this amounts to the one-way packet success rate of approximately 99 % (see histograms in Fig. 7 that exclude unsuccessful transmissions). Thus, when ACK requests are enabled, up to three data-link layer retransmissions are performed,[14] and we experimentally observed that no application-level retries are required beyond the three low-level protocol-provided retransmissions, in the case when a single industrial machine operates in isolation.

On the other hand, to increase network utilization, we emulated a number of additional machines communicating over the same wireless channel in physical vicinity (described in more detail in Section VII); we experimentally observed the one-way packet success rate of approximately 98 %. Thus, two application-level retries were sufficient to enable reliable exchange of events, ensuring correct operation. Intuitively, protocol-provided retries are issued in short bursts while application-level retransmissions incur significant delay; the channel is more likely to be continuously busy for a short period of time (e.g., occupied by other legitimate transmissions). Yet, an adversary may repeatedly deny network access to legitimate controllers preventing the system from progressing. Consequently, the modeled system does not satisfy Property 2, despite application-level retransmissions, unless DoS attacks can be detected and system halted (or other precautionary actions taken), using e.g., a separate secure channel.

From the operational perspective, every LC may implement a limited number of successive application-level retransmissions before declaring that it is under attack. For instance, if in our setup from Section VII, we limited the number of retransmissions to five, amounting to a theoretical one-way packet success rate of *eight nines* (if application-level retransmissions are assumed to be independent). To address this from the modeling perspective, we add an additional place where the transmitter's model transitions to, when application-level retries are exhausted (see Fig. 6(a)). Hence, we can verify that if infinite blocking of medium access is allowed, LCs may end up in the place `Pa_DoSdetect`. Conversely, if DoS attacks are limited to four consecutive channel access denials, Property 2 is satisfied. Note that immediate emergency halt of the machinery may not be possible if a secure communication channel is not available or the DoS attacks cannot be isolated from the network (e.g., using bus guardians).

2) Authenticating network flows: Traditional cryptographic techniques for ensuring integrity of network flows rely on signing packets with Message Authentication Codes (MAC) [17], and can be used to defend against spoofing attacks. In this setting, every transmission between LCs is signed by the transmitter using a secret key, and the signature is verified by the receiver; therefore, the attacker cannot tamper with the message payload, or else (s)he will be detected.

From the modeling perspective, introducing authentication can be modeled as an additional condition on the receiving transitions (in the controller models) where the received payload is compared to desired values; i.e., the MAC portion of the payload is compared to a secret value that cannot be altered (in the case of modification) or generated (in the case of spoofing attacks) by the attacker. Specifically, the transition `Tb_wfRx` in the LC_B model in Fig. 4(b) would feature an additional guard function on the `B_RxMAC` variable. Optionally, if the signature verification fails, a transition to a

[13] Romeo does not output statistics of the state space underlying the model.

[14] The XCVR model in Fig. 4(d) is compatible with these specifications.

place modeling intrusion detection reaction can be added as shown in Fig. 6(b); this is left to the application designer as reacting to detected intrusions is highly application-specific.

Using the developed framework, we verified that if non-authenticated transmissions are not allowed (i.e., authentication implemented), Properties 1 and 3 can be verified over our running example, under the condition that infinite denial of network access to LCs is not allowed, as previously discussed.

3) Acknowledgement Spoofing: Authenticating transmission does not affect ACKs as the data-link layer is responsible for ACK packets while MACs are added to the packet payload. In addition, non-encrypted sequence numbers, which are part of the packet frame, can be overheard by the attacker. Thus, valid ACKs can be generated on behalf of inactive (failed) LCs. Also, undelivered (i.e., intercepted) transmissions can be falsely acknowledged, even when authentication is used. This is a well-known shortcoming of data link layer ACKs [19], [70], and could be alleviated by application-level ACKs. Enforcing consensus over event-propagation in discrete event systems spans beyond the scope of this paper; yet, the presented modeling techniques can be utilized to model additional implemented protocols.

While this section introduced the general security-aware modeling aspects, with occasional focus on specific medium access techniques to avoid overly general discussions, in the following section we demonstrate the use of the presented framework on real-world industrial case studies.

VII. Case Studies: Industrial Manipulators

We consider a full physical implementation of a reconfigurable industrial pneumatic manipulator with a variable number of modules/degrees of freedom (DOF) controlled in a distributed fashion; i.e., one local controller per module/DOF. We demonstrate effectiveness of our framework on multiple module configurations (i.e., 2-DOF, 3-DOF).

A. 2-DOF Industrial Pneumatic Manipulator

The pneumatic industrial manipulator in the 2-DOF configuration is depicted in Fig. 8(a); two double-acting cylinders (denoted A and B) provide translational degrees of freedom, while the pneumatic gripper (denoted C) provides means of handling the workpiece. All actuation commands are issued by updating electrical signals `xp`, $\texttt{x} \in \{\texttt{a},\texttt{b},\texttt{c}\}$ which activate monostable dual control pneumatic valves.[15] Notice that signals are denoted with `x` while cylinders are denoted with X. Cylinders A an B are equipped with two proximity switches which allow position (i.e., fully retracted, fully extended) sensing. Signals corresponding to fully retracted (home) position are denoted `x0`, while fully extended (end) position signals are denoted `x1`. Additionally, the system contains a start switch whose corresponding signal is denoted by `st`.

CIPN-based models of three LCs are shown in Fig. 9. Initially, cylinders A and B are fully retracted, and gripper C released—in this state the manipulator is ready to begin its work cycle. The initial work cycle of the manipulator is started by pressing the start switch (`st==1`), after which operation is fully automated. First, cylinder B extends towards the workpiece picking position (due to actuation command `bp=1`). Once cylinder B reaches its end position (`b1==1`), gripper C is commanded gripping (`cp=1`). Controller B waits

[15] A control valve is the interface between the controller and the pneumatic cylinder; it converts the actuation signal from the controller into mechanical movement that controls flow of pressured air towards pneumatic cylinders.

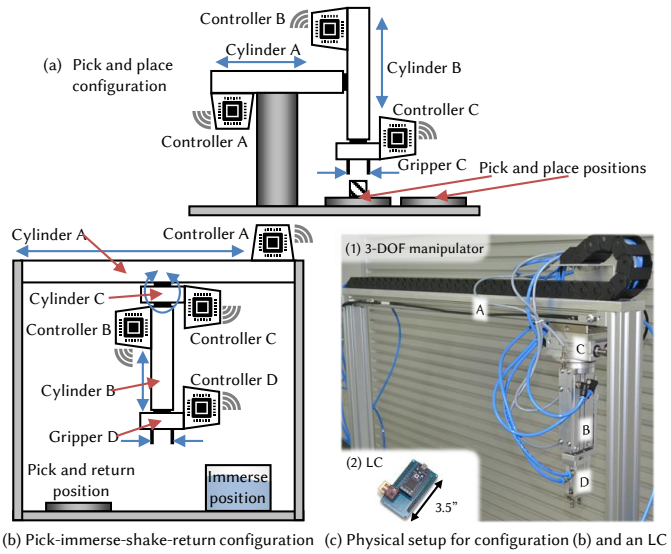

Fig. 8. Pneumatic manipulator in multiple configurations: (a) 2-DOF pick & place configuration; (b) 3-DOF pick-immerse-shake-return configuration; (c,1) upper portion of the physical setup of the configuration (b) shows cylinders; (c,2) low-cost ARM Cortex-M3-based networked controller; each physical component (cylinders and the gripper) are equipped with one LC.

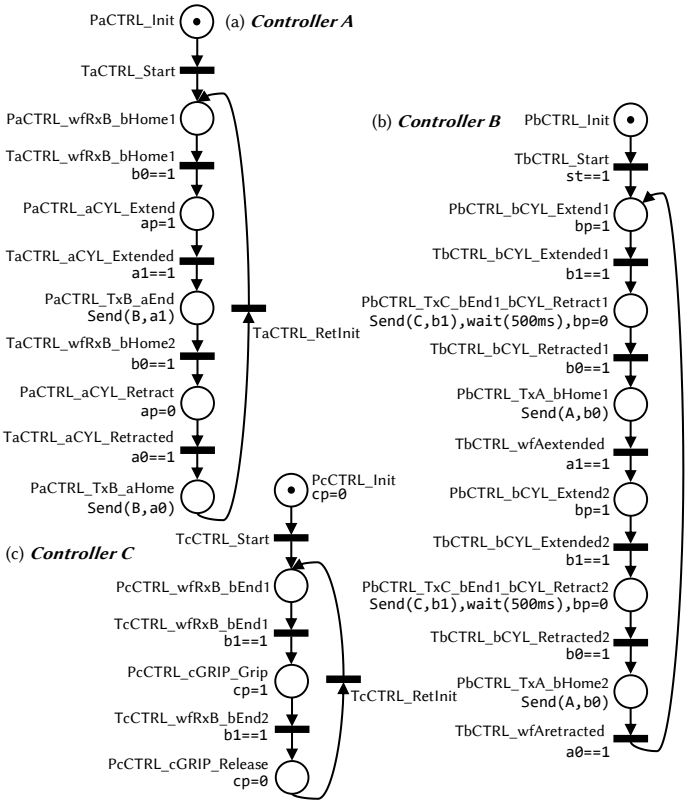

Fig. 9. CIPN-based distributed controller of a 2-DOF pneumatic manipulator.

for 500 ms for the part to be gripped.[16] Then, cylinder B retracts (due to command `bp=0`), and once it reaches home position (`b0==1`), cylinder A extends (due to command `ap=1`). After reaching its end position (`a1==1`), cylinder B extends towards the placing position (due to command `bp=1`). Once it reaches its end position (`b1==1`), gripper C is commanded release of the workpiece (command `cp=0`). 500 ms later, cylinder B retracts (`bp=0` followed by `b0==1`), after which cylinder A retracts (`ap=0` followed by `a0==1`). The manipulator returnees into its initial state, after which the next cycle is automatically executed. Signals (i.e., sensors outputs and actuator inputs) are allocated to LCs according to their physical proximity: $\{$`ap,a0,a1`$\}$ are mapped to controller A (i.e., LC_A), $\{$`bp,b0,b1,st`$\}$ to controller B (LC_B), and $\{$`cp`$\}$ to controller C (LC_C).

TPN models are obtained from these specifications as described in Section IV, but are omitted here due to their size. On the other hand, pneumatic cylinders are modeled as two-state plants with bounded, non-deterministic extending/retracting times obtained from experimental measurements. We extract timing parameters (i.e., bounds on time-to-transmit, time-to-acknowledge, and back-off timing) from experimental measurements—histograms for $10,000$ messages are shown in Fig. 7, for the employed low-cost ARM Cortex-M3-based controllers equipped with an IEEE 802.15.4-compliant transceiver. On the other hand, we obtain transceiver-related timings (e.g., back-off time during clear channel assessment) from the radio specifications [67]. While we verified a large number of safety and liveness properties for this setup, we illustrate verification and security patching on a more complex 3-DOF setup.

B. 3-DOF Industrial Pneumatic Manipulator

A 3-DOF configuration of the described manipulator is shown in Fig. 8(b). The additional rotational DOF, provided by cylinder C, introduces an additional LC and increases the complexity of the LC coordination. This configuration may be used to prepare workpieces for painting by immersing them into a pool with cleaning solution, and returning them to the pick-up position for further processing by another machine.

Fig. 8(c,1) shows the physical setup for this configuration; the upper portion of the manipulator is shown such that cylinders are visible. Fig. 8(c,2) shows the low-cost ARM Cortex-M3-based LC with the corresponding IEEE 802.15.4 transceiver. While the models are more complex than in the 2-DOF case, they are semantically similar and thus omitted. Fig. 10(a) shows event timing—i.e., states of all sensing and actuation signals, for one sample pick-immerse-shake-return run; messages exchanged by LCs are denoted with blue arrows originating at the source event and terminating at the triggered event. Among the many safety liveness and QoC properties, we illustrate verification on the following examples.

Property 4: Gripper D is always gripped before cylinder B picks the workpiece; formally captured as, `AG(M(PdGRIP_Gripped)==1 and M(PbCTRL_bCYL_Retract1)==1)`.

[16]The gripper C does not have end position sensing due to size constraints; thus a timed delay is used to permit secure gripping/releasing of the workpiece.

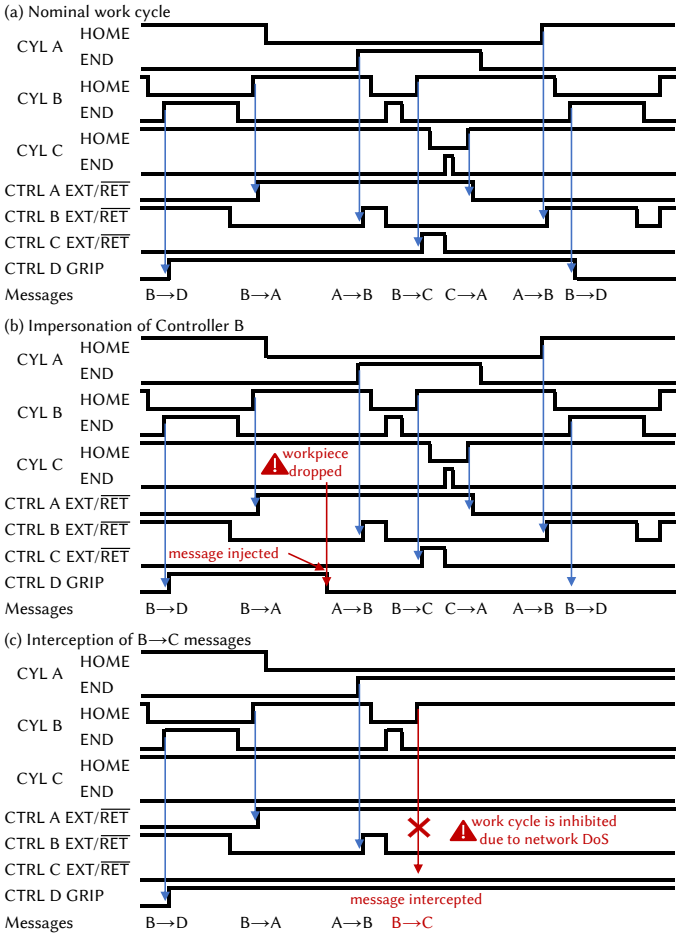

Fig. 10. Sensing/actuation signal timings for a nominal pick & place run (a), a run where a signal injection is performed resulting in a dropped workpiece (b), and a run where progress is inhibited due to a DoS attack (c). Messages exchanged by LCs are marked with blue arrows. X axis is unlabeled as the speed of the workcycle can be controlled by regulating air pressure in the system and is thus not crucial.

Property 5: A workpiece is eventually processed, once the work cycle is started. Formally, `M(PbCTRL_bCyl_Extend1)==1--> (M(PcCTRL_cGRIP_Release)==1.`

When no security mechanisms are employed, we verified violation of these properties. Property 4 is violated due to a possible impersonation attack at the gripper controller; an attacker may send the command to release the workpiece before it was returned to the return position. Fig. 10(b) shows signal timings acquired on a sample cycle run in which the workpiece is dropped due to a maliciously injected command to release the gripper (potentially causing mechanical damage to the workpiece and/or the manipulator).

However, if transmissions are authenticated, and the model adjusted correspondingly as described in Section VI-A2, this vulnerability is alleviated. We applied a software security patch by including the *mbed TLS* (Secure Sockets Layer) library that our IIoT controllers are fully compatible with. Signing a 128 bit message authentication code over one transmitted signal incurs computational overhead of ~ 100 μs on the employed low-cost ARM Cortex-M3-based LCs; this

practically negligibly slows down manipulator's work cycle, while providing security guarantees. Hence, Property 4 is satisfied following this security patch.

Property 5 is violated due to the possibility of a DoS attack that infinitely delays progress. From the model's perspective, this attack does not cause a deadlock—while the *physical* process is stalled, the *cyber* process is in fact livelocked reattempting to access the channel (i.e., same places are revisited and same transitions fire infinitely often). Fig. 10(c) shows signal timings acquired on a sample run where a DoS attack is launched by enabling carrier transmission on the attacker's transceiver, in order to jam messages after the workpiece was picked up from the immersion pool. As described in Section VI-A1, wireless control nodes can keep track of unsuccessful medium access attempts, and promptly halt operation when a DoS attack is detected. In such cases, distributing the information about DoS detection requires a secure channel, which we do not consider in this work.

VIII. Conclusion and Discussion

In this paper, we have introduced a framework for security analysis of distributed sequential control systems captured by CIPN-based models. As CIPNs do not support verification of safety properties in the presence of attacks, our approach transforms control models into TPNs that inherently enable this verification by supporting non-determinism in transition times as well as transition firing; this imposes minimal assumptions on adversarial actions. We have shown how a model of a network-based attacker can be integrated into the non-deterministic communication channel model, and verified violation of safety properties in presence of attacks.

Additionally, we have shown how verification results can be used to pinpoint vulnerabilities in control software implementation, suggesting security patches to alleviate the impact of the attacks on control performance. We have also provided the loop back to the modeling stage, enabling re-verification of properties that are now satisfied due to the use of the appropriate security mechanisms. Finally, we have evaluated our framework on a real-world industrial case study.

Note that the runtime effects such as utilization of communication channels and mechanical wear may affect parameters of developed models (e.g., timing bounds in TPN). Thus, the presented system can be combined with runtime monitoring that evaluates whether real-time process and communication channel measurements comply with the aforementioned plant and channel models, raising alarm if potential violations are detected (e.g., similar to [47]). Finally, providing resilience of low-level distributed controllers is only a part of securing IoT-based Industry 4.0 systems. Integrating such resilient subsystems with enterprise-level resilience mechanisms (e.g., data-driven monitoring) will be a part of the future work.

Acknowledgment

This work is sponsored in part by the ONR under agreements N00014-17-1-2012, N00014-17-1-2504, and N00014-20-1-2745, the NSF CNS-1652544 grant, and by the Science Fund of the Republic of Serbia, grant No. 6523109, AI-MISSION 4.0.

References

[1] Z. Jakovljevic, V. Majstorovic, S. Stojadinovic, S. Zivkovic, N. Gligorijevic, and M. Pajic, "Cyber-Physical Manufacturing Systems (CPMS)," in *Proc. of 5th Int. Conf. on Advanced Manufacturing Engineering and Technologies*. Springer International, 2017, pp. 199–214.

[2] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster, *Recommendations for implementing the strategic initiative INDUSTRIE 4.0*. Forschungsunion, 2013.

[3] H. ElMaraghy, G. Schuh, W. ElMaraghy, F. Piller, P. Schönsleben, M. Tseng, and A. Bernard, "Product variety management," *CIRP Annals - Manufacturing Technology*, vol. 62, no. 2, pp. 629 – 652, 2013.

[4] Y. Wang, W. Zhao, and W. X. Wan, "Needs-based product configurator design for mass customization using hierarchical attention network," *IEEE Transactions on Automation Science and Engineering*, pp. 1–10, 2020.

[5] E. Trunzer, A. Calà, P. Leitão, M. Gepp, J. Kinghorst, A. Lüder, H. Schauerte, M. Reifferscheid, and B. Vogel-Heuser, "System architectures for industrie 4.0 applications: Derivation of a generic architecture proposal," *Production Engineering*, vol. 13, no. 3-4, pp. 247–257, 2019.

[6] Y. Koren, X. Gu, and W. Guo, "Reconfigurable manufacturing systems: Principles, design, and future trends," *Frontiers of Mechanical Engineering*, vol. 13, no. 2, pp. 121–136, 2018.

[7] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (iiot): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, 2018.

[8] K. Ding and P. Jiang, "Rfid-based production data analysis in an iot-enabled smart job-shop," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 128–138, 2018.

[9] Industrial Internet Consortium, "Industrial internet reference architecture," 2015. [Online]. Available: https://www.iiconsortium.org/IIRA-1-7-ajs.pdf

[10] S. Guellouz, A. Benzina, M. Khalgui, G. Frey, Z. Li, and V. Vyatkin, "Designing efficient reconfigurable control systems using iec61499 and symbolic model checking," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1110–1124, 2019.

[11] J. Otto, B. Vogel-Heuser, and O. Niggemann, "Automatic parameter estimation for reusable software components of modular and reconfigurable cyber-physical production systems in the domain of discrete manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 275–282, 2018.

[12] F. Muram, M. Javed, H. Hansson, and S. Punnekkat, "Dynamic reconfiguration of safety-critical production systems," vol. 2020-December, 2020, pp. 120–129.

[13] D.-Y. Kim, J.-W. Park, S. Baek, K.-B. Park, H.-R. Kim, J.-I. Park, H.-S. Kim, B.-B. Kim, H.-Y. Oh, K. Namgung, and W. Baek, "A modular factory testbed for the rapid reconfiguration of manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 31, no. 3, pp. 661–680, 2020.

[14] L. Xu, E. Xu, and L. Li, "Industry 4.0: State of the art and future trends," *International Journal of Production Research*, vol. 56, no. 8, pp. 2941–2962, 2018.

[15] Y. Wang, A. K. Bozkurt, and M. Pajic, "Attack-resilient supervisory control of discrete-event systems," *rXiv:1904.03264 [cs.FL]*, Apr. 2019.

[16] M. Pajic, J. Weimer, N. Bezzo, O. Sokolsky, G. J. Pappas, and I. Lee, "Design and implementation of attack-resilient cyberphysical systems: With a focus on attack-resilient state estimators," *IEEE Control Systems*, vol. 37, no. 2, pp. 66–81, April 2017.

[17] Y. Xiao, H.-H. Chen, B. Sun, R. Wang, and S. Sethi, "Mac security and security overhead analysis in the ieee 802.15. 4 wireless sensor networks," *EURASIP J. on Wireless Communications and Networking*, no. 2, pp. 81–81, 2006.

[18] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 1, pp. 112–125, 2007.

[19] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," in *Proc. of the First IEEE Int. Workshop on Sensor Network Protocols and Applications*, 2003, pp. 113–127.

[20] Z. Song, A. Skuric, and K. Ji, "A recursive watermark method for hard real-time industrial control system cyber-resilience enhancement," *IEEE Transactions on Automation Science and Engineering*, pp. 1–14, 2020.

[21] M. Mohsin, Z. Anwar, G. Husari, E. Al-Shaer, and M. A. Rahman, "IoTSAT: A formal framework for security analysis of the internet of things (IoT)," *2016 IEEE Conference on Communications and Network Security (CNS)*, pp. 180–188, Oct 2016.

[22] S. Zahra, M. Alam, Q. Javaid, A. Wahid, N. Javaid, S. U. R. Malik, and M. Khurram Khan, "Fog computing over iot: A secure deployment and formal verification," *IEEE Access*, vol. 5, pp. 27 132–27 144, 2017.

[23] M. Houimli, L. Kahloul, and S. Benaoun, "Formal specification, verification and evaluation of the mqtt protocol in the internet of things," in *2017 International Conference on Mathematics and Information Technology (ICMIT)*, 2017, pp. 214–221.

[24] Z. Jakovljevic, V. Lesi, S. Mitrovic, and M. Pajic, "Distributing sequential control for manufacturing automation systems," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 4, pp. 1586–1594, 2019.

[25] Z. He, Z. Ma, Z. Li, and A. Giua, "Parametric transformation of timed weighted marked graphs: applications in optimal resource allocation," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, pp. 179–188, 2020.

[26] B. Huang, M. Zhou, A. Abusorrah, and K. Sedraoui, "Scheduling robotic cellular manufacturing systems with timed petri net, a* search, and admissible heuristic function," *IEEE Transactions on Automation Science and Engineering*, 2020.

[27] L. Pan, B. Yang, J. Jiang, and M. Zhou, "A time petri net with relaxed mixed semantics for schedulability analysis of flexible manufacturing systems," *IEEE Access*, vol. 8, pp. 46 480–46 492, 2020.

[28] G. Gardey, D. Lime, M. Magnin *et al.*, "Romeo: A tool for analyzing time petri nets," in *International Conference on Computer Aided Verification*. Springer, 2005, pp. 418–423.

[29] M. Cuka, D. Elmazi, K. Bylykbashi, E. Spaho, M. Ikeda, and L. Barolli, "Effect of node centrality for iot device selection in opportunistic networks: A comparison study," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 21, p. e4790, 2018.

[30] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric iot," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2018.

[31] N. Rashid, J. Wan, G. Quiros, A. Canedo, and M. A. A. Faruque, "Modeling and simulation of cyberattacks for resilient cyber-physical systems," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, Aug 2017, pp. 988–993.

[32] N. Rashid, G. Quirós, and M. A. A. Faruque, "A survivability-aware cyber-physical systems design methodology," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, Aug 2019, pp. 848–853.

[33] Y. Zhang, Y. Shen, H. Wang, J. Yong, and X. Jiang, "On secure wireless communications for iot under eavesdropper collusion," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 3, pp. 1281–1293, July 2016.

[34] Z. B. Celik, P. McDaniel, and G. Tan, "Soteria: Automated iot safety and security analysis," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018, pp. 147–158.

[35] Z. B. Celik, G. Tan, and P. D. McDaniel, "IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT," in *Network and Distributed System Security Symposium (NDSS)*, 2019.

[36] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Model Checking*. Springer, 2018, pp. 305–343.

[37] Z. Jakovljevic, V. Lesi, and M. Pajic, "Attacks on distributed sequential control in manufacturing automation," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 775–786, 2021.

[38] G. Fortino, F. Messina, D. Rosaci, and G. Sarne, "Resiot: An iot social framework resilient to malicious activities," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 5, pp. 1263–1278, 2020.

[39] Z. Zhang, D. Yue, C. Dou, and H. Zhang, "Multiagent system-based integrated design of security control and economic dispatch for interconnected microgrid systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 4, pp. 2101–2112, 2021.

[40] E. A. Alzalab, Z. Yu, N. Wu, and H. Kaid, "Fault-recovery and repair modeling of discrete event systems using petri nets," *IEEE Access*, vol. 8, pp. 170 237–170 247, 2020.

[41] J. P. McDermott, "Attack net penetration testing," in *New Security Paradigms Workshop (NSPW)*, 2000, pp. 15–21.

[42] Dalton, Mills, Colombi, and Raines, "Analyzing attack trees using generalized stochastic petri nets," in *2006 IEEE Information Assurance Workshop*, June 2006, pp. 116–123.

[43] T. M. Chen, J. C. Sanchez-Aarnoutse, and J. Buford, "Petri net modeling of cyber-physical attacks on smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 741–749, Dec 2011.

[44] J. Steffan and M. Schumacher, "Collaborative attack modeling," in *Proceedings of the 2002 ACM Symposium on Applied Computing*, ser. SAC '02. ACM, 2002, pp. 253–259.

[45] R. Mitchell and I. Chen, "Effect of intrusion detection and response on reliability of cyber physical systems," *IEEE Transactions on Reliability*, vol. 62, no. 1, pp. 199–210, March 2013.

[46] A. Ashok, A. Hahn, and M. Govindarasu, "Cyber-physical security of wide-area monitoring, protection and control in a smart grid environment," *Journal of Advanced Research*, vol. 5, no. 4, pp. 481–489, 2014.

[47] V. Lesi, Z. Jakovljevic, and M. Pajic, "Reliable industrial iot-based distributed automation," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, ser. IoTDI '19. ACM, 2019, pp. 94–105.

[48] M. P. Fanti, A. M. Mangini, and W. Ukovich, "Fault detection by labeled petri nets in centralized and distributed approaches," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 2, pp. 392–404, April 2013.

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

[50] A. K. Bozkurt, Y. Wang, and M. Pajic, "Secure Planning Against Stealthy Attacks via Model-Free Reinforcement Learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[51] A. Zimmermann and G. Hommel, "Towards modeling and evaluation of etcs real-time communication and operation," *Journal of Systems and Software*, vol. 77, no. 1, pp. 47 – 54, 2005.

[52] M. Diaz, "Modeling and analysis of communication and cooperation protocols using petri net based models," *Computer Networks*, vol. 6, no. 6, pp. 419 – 441, 1982.

[53] R. David and H. Alla, "Petri nets for modeling of dynamic systems: A survey," *Automatica*, vol. 30, no. 2, pp. 175–202, 1994.

[54] ——, *Discrete, continuous, and hybrid petri nets (2nd edition)*, 2010.

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

[56] M. Sogbohossou and A. Vianou, "Formal modeling of grafcets with time petri nets," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1978–1985, Sep. 2015.

[57] N. Wightkin, U. Buy, and H. Darabi, "Formal modeling of sequential function charts with time petri nets," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 2, pp. 455–464, 2011.

[58] F. Cassez and O. H. Roux, "Structural translation from time petri nets to timed automata," *Journal of Systems and Software*, vol. 79, no. 10, pp. 1456–1468, 2006.

[59] S. Donatelli, "Superposed stochastic automata: a class of stochastic petri nets with parallel solution and distributed state space," *Performance Evaluation*, vol. 18, no. 1, pp. 21 – 36, 1993.

[60] L. Popova-Zeugmann, "Time petri nets," in *Time and Petri nets*. Springer, 2013, pp. 31–137.

[61] J. Wang, Y. Deng, and G. Xu, "Reachability analysis of real-time systems using time petri nets," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 30, no. 5, pp. 725–736, 2000.

[62] M. dos Santos Soares, S. Julia, and J. Vrancken, "Real-time scheduling of batch systems using petri nets and linear logic," *Journal of Systems and Software*, vol. 81, no. 11, pp. 1983 – 1996, 2008.

[63] D. Lime and O. H. Roux, "A translation based method for the timed analysis of scheduling extended time petri nets," in *25th IEEE International Real-Time Systems Symposium*, Dec 2004, pp. 187–196.

[64] O. H. Roux and A.-M. Déplanche, "A t-time petri net extension for real time-task scheduling modeling," *European Journal of Automation*, vol. 36, no. 7, pp. 973–987, 2002.

[65] S. Hong and S. Lim, "Analysis of attack models via unified modeling language in wireless sensor networks: A survey study," in *2010 IEEE International Conference on Wireless Communications, Networking and Information Security*, 2010, pp. 692–696.

[66] A. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[67] Microchip Technology Inc., "MRF24J40MA 2.4 GHz IEEE Std. 802.15.4™RF Transceiver Module," 2008.

[68] B. Berthomieu, P.-O. Ribet, and F. Vernadat, "The tool tina–construction of abstract state spaces for petri nets and time petri nets," *International journal of production research*, vol. 42, no. 14, pp. 2741–2756, 2004.

[69] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[70] N. Sastry and D. Wagner, "Security considerations for IEEE 802.15. 4 networks," in *Proceedings of the 3rd ACM workshop on Wireless security*, 2004, pp. 32–42.

Vuk Lesi received his B.Sc. degree in electrical and computer engineering from the University of Belgrade, Serbia, in 2015, and his Ph.D. in electrical and computer engineering from Duke University, Durham, North Carolina, in 2019. He is currently a research scientist at Intel Labs, where he focuses on developing safe and secure cyber-physical systems.

Dr. Lesi received multiple recognitions including the Best Paper Award at the 2017 ACM SIGBED International Conference on Embedded Software. He has been contributing as a technical program committee member to the IEEE Technical Committee on Real-Time Systems as well as the IEEE Industrial Electronics Society, IEEE Council on Electronic Design Automation, and the ACM Special Interest Group on Applied Computing. His research interests span security-aware autonomous systems, distributed industrial automation, embedded, real-time, and safety-critical cyber-physical systems.

Zivana Jakovljevic (M'18) received the Dipl. Ing., M.Sc. and Ph.D. degrees in mechanical engineering from the Faculty of Mechanical Engineering, University of Belgrade, Serbia, in 1999, 2004 and 2010, respectively.

She is currently Full Professor and Head of Laboratory for Manufacturing Automation at Faculty of Mechanical Engineering, University of Belgrade, Serbia, where she was the member of academic staff since 2001. Her research interests include intelligent manufacturing systems, cyber physical systems, industrial internet of things, distributed control, 3D vision systems in manufacturing automation, machine learning and non-stationary signal processing.

Miroslav Pajic (S'06-M'13-SM'19) received the Dipl. Ing. and M.S. degrees in electrical engineering from the University of Belgrade, Serbia, in 2003 and 2007, respectively, and the M.S. and Ph.D. degrees in electrical engineering from the University of Pennsylvania, Philadelphia, in 2010 and 2012, respectively.

He is the Dickinson Family Associate Professor in Electrical and Computer Engineering Department at Duke University, with a secondary appointment in the Computer Science Department. His research focuses on the design and analysis of high-assurance cyber-physical systems with varying levels of autonomy and human interaction.

Dr. Pajic received 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 in the ACM Transactions on Computing for Healthcare and was a co-Chair of the 2019 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'19).