Supervisory Control of Petri Nets in the Presence of Replacement Attacks

Dan You, ShouGuang Wang, MengChu Zhou, *Fellow, IEEE*, and Carla Seatzu

***Abstract*—This work addresses the robust control problem of discrete event systems assuming that replacement attacks may occur, thus making it appear that an event that has occurred looks like another event. In particular, we assume that this is done by tampering with the sensor-readings in the sensor communication channel. Specifically, we use Petri nets as the reference formalism to model the plant and assume a control specification in terms of a Generalized Mutual Exclusion Constraint. We propose three different methods to derive a control policy that is robust to the possible replacement attacks. The first two methods lead to an optimal (i.e., maximally permissive) policy but are computationally inefficient when applied to large-size systems. On the contrary, the third method computes a policy more efficiently, and reveals more easily implementable in practice. However, this is done at the expense of optimality.**

***Index Terms*—Discrete event systems, attacks, Petri nets, supervisory control.**

I. Introduction

The supervisory control problem of discrete event systems (DESs) has been extensively studied in the literature [5]. Its solution requires the design of a supervisor that enforces a given control specification by appropriately restricting the behavior of the system. This is realized by disabling some events depending on the current observation. In a realistic scenario, the supervisor observes the occurrence of events by getting the readings from sensors and disables some of the events by issuing commands to control the related actuators. The architecture of such a closed-loop control system is shown in Fig. 1.

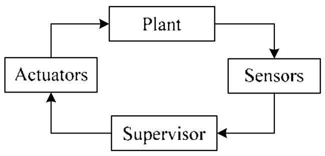

Fig. 1. Closed-loop control system

This work is supported in part by Project RASSR05871 MOSIMA financed by Region Sardinia, FSC 2014-2020, annuity 2017, Subject area 3, Action Line 3.1, in part by the Zhejiang Natural Science Foundation under Grant LQ20F020009, in part by the Zhejiang Provincial Key Laboratory of New Network Standards and Technologies under Grant 2013E10012, and in part by the Public technology research plan of Zhejiang Province under Grant LGJ21F030001. (*Corresponding author: ShouGuang Wang*)

D. You and C. Seatzu are with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, Italy (e-mail: youdan000@hotmail.com; carla.seatzu@unica.it).

S. G. Wang is with School of Information & Electronic Engineering, Zhejiang Gongshang University, Hangzhou, 310018, China (e-mail: wsg5000@hotmail.com).

M. C. Zhou is with the Helen and John C. Hartmann Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Nowadays, communication networks are increasingly used to connect individual modules in the closed-loop control system. This makes the system vulnerable to various network attacks. As a consequence, it could happen that the sensor-readings and/or the control commands transmitted in the communication network are compromised and appropriately altered by some malicious attacker. Consequently, it is necessary to reconsider the supervisory control problem of DESs in the presence of attacks.

In the last two decades several significant contributions have been provided in this framework. Thorsley and Teneketzis [12] focus on finding a control specification whose corresponding supervisor is capable of preventing damages caused by attacks in the actuator channel. Focusing on a particular class of attacks, Carvalho *et al.* [3] and Lima *et al.* [6-8] develop defense strategies that detect intrusions on-line and protect the plant from damages by activating an appropriate security policy once an intrusion is detected. Agarwal *et al.* [1, 2] also study intrusion detection for other special classes of attacks. Essentially, these studies are similar to those on fault detection and diagnosis [18].

A series of significant contributions have also been proposed with the goal of deriving a supervisory control law that is robust to attacks. Wakaiki *et al.* [14] propose a supervisor, modeled as a finite-state automaton, which is robust to *replacement-removal attacks*, namely attacks where the sensor signals produced by certain events can be replaced by the sensor signals produced by other events, or can be canceled. It is worth noting that a DES vulnerable to attacks that modify the sensor-readings can be modelled as a DES with nondeterministic partial observations. The supervisory control problem is studied in [13, 15, 17]. Given a language specification, [13, 15] basically investigate conditions under which a supervisor exists exactly enforcing the given specification, while [17] develops a supervisor synthesis method based on modeling transformation.

Also, several recent works [11, 19, 20] tackle intelligent attacks, which inflict damages on the plant without being detected. Su [11] first constructs the supremal intelligent attack strategy from the viewpoint of an attacker and then seeks for a robust supervisor. Zhang *et al.* [19, 20] consider state estimation under partial observation in the presence of attacks. A tool is proposed that enables the system operator to easily establish how the estimation of the set of states consistent with the uncorrupted observation is related with the set of states consistent with the corrupted observation.

In this manuscript, we study the robust control problem of DESs in the presence of *replacement attacks* that may disguise the occurrence of certain events as the occurrence of other events. Differently from almost all the above papers that use finite-state automata as the reference formalism, this paper adopts Petri nets (PNs). The major advantage deriving from this is that the proposed approaches may also be applied to unbounded DESs [9]. Moreover, we assume that control specifications are given in terms of Generalized Mutual Exclusion Constraints (GMECs) [4]. This choice is typical in the PN framework. Indeed, GMECs allow one to use linear integer programming and the theory of monitor places by which GMECs may be enforced very easily and the maximal permissiveness is guaranteed in the case of controllable and observable transitions. In more detail, we propose three different methods. The first method provides an optimal (namely, maximally permissive) control policy based on the enumeration of markings consistent with the current observation. The second method derives an optimal policy based on constructing a monitor-controlled PN system, which still requires marking enumeration. The third method computes a control policy with timely response even for large-size PN systems at the expense of optimality.

We note that [14] and [17] are closely related to our work. Both of them consider the supervisor synthesis problem under attack. Specifically, their focus is that of synthesizing a supervisor off-line represented by a finite-state automaton. Once such a supervisor is synthesized, little on-line computation is needed. However, computing such a supervisor requires an exhaustive reachability analysis, which makes such approaches hardly applicable to DESs with a large number of states and unfeasible in the case of DESs with an unbounded state space. In contrast, this work develops on-line control policies that are applicable to both bounded and unbounded systems in the formalism of PNs. On the other side, [14] and [17] present the advantage of dealing with more general attacks.

As a conclusion, the main merit of this paper is that of providing a first contribution to the investigation of the problem of supervisory control under attack using PNs.

II. Preliminaries

A. Petri Nets

A *Petri net* (PN) [10] is a four-tuple $N=(P, T, F, W)$ where P and T are nonempty, finite and disjoint sets. P is the set of *places* represented by circles and T is the set of *transitions* represented by bars. $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation* represented by directed arcs. W: $(P \times T)\cup(T \times P)\rightarrow\mathbb{N}=\{0, 1, 2, \ldots\}$ assigns a weight to each arc such that $W(x, y)>0$ if $(x, y)\in F$, and $W(x, y)=0$ otherwise, where $x, y\in P\cup T$.

The *incidence matrix* of N is a matrix $[N]$: $P\times T\rightarrow\mathbb{Z}$ indexed by P and T such that $[N](p, t) =W(t, p)-W(p, t)$, where $\mathbb{Z}$ is the set of integers.

Given a node $x\in P\cup T$, ${}^\bullet x =\{y\in P\cup T \mid (y, x)\in F\}$ is the set of *inputs* of x, while $x^\bullet=\{y\in P\cup T \mid (x, y)\in F\}$ is the set of *outputs* of x. Furthermore, $\forall X\subseteq P\cup T$, ${}^\bullet X=\bigcup_{x\in X}{}^\bullet x$ and $X^\bullet=\bigcup_{x\in X}x^\bullet$.

A *marking* or *state* of N is a vector m: $P\rightarrow\mathbb{N}$, where $m(p)$ is the number of *tokens* in place p at marking m. A transition $t\in T$ is *enabled* at a marking m, denoted as $m[t\rangle$, if $m(p) \geq W(p, t)$, $\forall p\in {}^\bullet t$. The set of transitions enabled at m is denoted by $En(m)$. Moreover, the notation is extended to a marking set $\mathcal{M}$ such that $En(\mathcal{M})= \bigcup_{m\in\mathcal{M}}En(m)$. A transition t can *fire* at m if t is enabled at m. We use $m[t\rangle m'$ to denote that a marking m' is reached from m by firing t and it holds that $m'(p)=m(p)-W(p, t)+W(t, p)$, $\forall p\in P$. We also use m_t to denote the marking reached by firing t at m. Given a transition sequence $\alpha=t_1t_2\ldots t_k\in T^*$, α is enabled at m, denoted as $m[\alpha\rangle$, if there exist markings $m_1, m_2, \ldots, m_{k-1}$ such that $m[t_1\rangle m_1[t_2\rangle m_2[t_3\rangle \ldots m_{k-1}[t_k\rangle$. We use $m[\alpha\rangle m_k$ to denote that m_k is reached by firing α at m. We also use m_α to denote the marking reached by firing α at m.

The *initial marking* of a PN is denoted as m_0 and (N, m_0) is called a *net system* with initial marking m_0. We use $L(N, m_0)$ to denote the set of all transition sequences of N that are enabled at m_0, i.e., $L(N, m_0)=\{\alpha\in T^* \mid m_0[\alpha\rangle\}$, called the *language* of (N, m_0), and use $R(N, m_0)$ to denote the set of all reachable markings of N from m_0, i.e., $R(N, m_0)=\{m \mid \exists\alpha\in T^*, m_0[\alpha\rangle m\}$.

A PN system (N, m_0) is *bounded* if the number of tokens in each place p does not exceed a finite number $B\in\mathbb{N}^+=\{1, 2, \ldots\}$ for any marking $m\in R(N, m_0)$. Otherwise, it is *unbounded*.

Given a transition sequence $\alpha\in T^*$, $\mathcal{P}_r(\alpha)=\{\alpha'\in T^* \mid \exists\alpha''\in T^*$ s.t. $\alpha=\alpha'\alpha''\}$ is the set of all *prefixes* of α.

B. Control Policies

When dealing with supervisory control problems it is fundamental to establish which transitions are controllable/uncontrollable and which ones are observable/unobservable. In simple words, controllable transitions are those that can be disabled by the supervisor, while uncontrollable transitions may never be disabled by the supervisor; observable transitions if fired may be seen by the supervisor, while unobservable transitions if fired cannot be seen by the supervisor. In this paper, we assume that all transitions are observable and controllable.

A *control policy* [5] of a PN system (N, m_0) is a function

$$\rho\colon L_o(N, m_0)\rightarrow 2^T,$$

where $L_o(N, m_0)$ is the set of all possible observations during the evolution of the system. Specifically, ρ associates with an observation the set of transitions that should be disabled. Hence, we call $\rho(\delta)$ the *disabled set* associated with the observation $\delta\in L_o(N, m_0)$. Clearly, $L_o(N, m_0)$ coincides with $L(N, m_0)$ in the case that all transitions are observable.

The system (N, m_0) supervised under policy ρ is denoted as $(N, m_0)|_\rho$. The reachability set, the language, and the observed language of $(N, m_0)|_\rho$ are denoted by $R(N, m_0)|_\rho$, $L(N, m_0)|_\rho$ and $L_o(N, m_0)|_\rho$, respectively.

Given a PN system (N, m_0), policy ρ_1 is said to be *more permissive* than policy ρ_2, denoted as $\rho_1\succ\rho_2$, if

$$L(N, m_0)|_{\rho_1} \supset L(N, m_0)|_{\rho_2}.$$

Moreover, given a set of legal states $\mathcal{Q}\subseteq\mathbb{N}^{|P|}$, a policy ρ is said to be *acceptable* if $R(N, m_0)|_\rho \subseteq \mathcal{Q}$; and is said to be *optimal* (*maximally permissive*) if 1) $R(N, m_0)|_\rho\subseteq \mathcal{Q}$; and 2) $\forall\rho' \succ\rho$, $R(N, m_0)|_{\rho'} \not\subseteq \mathcal{Q}$.

We notice that the nomenclature used in this paper is summarized in [21] where some supplementary material is reported, which is omitted here for sake of brevity.

III. Problem Statement

In this paper, we study the supervisory control problem of PNs in the presence of attacks. In particular, we consider a state specification defined by a *Generalized Mutual Exclusion Constraint* (GMEC) and the attacks defined as *replacement attacks*. In this section, we firstly recall the related notion of GMECs, then introduce replacement attacks and finally provide the problem formulation.

A. GMECs

A *Generalized Mutual Exclusion Constraint* (GMEC) [4] is defined as a pair (ω, k), where ω: $P\to\mathbb{N}$ is a weight vector and $k\in\mathbb{N}$, which identifies the *legal marking set*:

$$\mathcal{L}_{(\omega, k)}=\{m\in\mathbb{N}^{|P|} \mid \omega\cdot m\leq k\}.$$

Moreover, we denote $\varpi=\omega\cdot[N]$. Given a transition t, it holds that $\varpi(t)=\omega\cdot m'-\omega\cdot m$, for any pair of markings m and m' such that m' is the marking reached from m by firing t. In other words, $\varpi(t)$ is equal to the variation of the ω-weighted sum of tokens in the net caused by the firing of t.

Given a set of GMECs $\Omega=\{(\omega_1, k_1), (\omega_2, k_2), \ldots, (\omega_n, k_n)\}$, where $n\in\mathbb{N}^+$, the *conjunction* of GMECs in Ω is denoted as $\wedge\Omega$, which defines the *legal marking set*: $\mathcal{L}_{\wedge\Omega}=\bigcap_{(\omega,k)\in\Omega}\mathcal{L}_{(\omega,k)}$.

B. Replacement attacks

In a replacement-attack scenario, an intruder has the ability to disguise the firing of a transition as another transition by tampering with the sensor-readings transmitted in a sensor communication channel. Formally, we define a *replacement attack* as $\mathcal{A}\in 2^{T\times T}$, i.e., $\mathcal{A}$ is a set of pairs of transitions, characterizing the capability of an intruder to disguise transitions. Specifically, each element in $\mathcal{A}$ denotes the pair (*transition that has fired*, *transition that appears to have fired*). Note that we assume that $\mathcal{A}$ does not include pairs with identical transitions.

Given a replacement attack $\mathcal{A}$ and a transition t, we denote

$$A(t)=\{t\}\cup\{t' \mid (t, t')\in \mathcal{A}\} \text{ and } A^{-1}(t)=\{t\}\cup\{t' \mid (t', t)\in \mathcal{A}\}.$$

In words, $A(t)$ is the set of all possible observations when transition t fires, while $A^{-1}(t)$ is the set of transitions whose firing may have produced the observation t.

Example 1: Consider a PN with $T=\{t_1\text{-}t_5\}$ and a replacement attack $\mathcal{A}=\{(t_1, t_2), (t_2, t_3), (t_2, t_4), (t_3, t_4)\}$. It holds: $A(t_1)=\{t_1, t_2\}$, $A(t_2)=\{t_2, t_3, t_4\}$, $A(t_3)=\{t_3, t_4\}$, $A(t_4)=\{t_4\}$, $A(t_5)=\{t_5\}$. Moreover, $A^{-1}(t_1)=\{t_1\}$, $A^{-1}(t_2)=\{t_1, t_2\}$, $A^{-1}(t_3)=\{t_2, t_3\}$, $A^{-1}(t_4)=\{t_2, t_3, t_4\}$, and $A^{-1}(t_5)=\{t_5\}$. ■

C. Problem Formulation

In this paper we study the following control problem.

Problem 1: Given a PN system (N, m_0) vulnerable to a replacement attack $\mathcal{A}$ and a GMEC (ω, k), design a robust control policy ρ that enforces the GMEC (ω, k).

Like most papers dealing with the supervisory control problem, we assume that the initial marking m_0 is legal.

Initially, Problem 1 is investigated with the goal of determining an optimal control policy under attack. Then, such a requirement is relaxed, in order to provide a more efficient approach in terms of computational complexity.

IV. On-line Optimal Control Policy

In this section, we present an on-line control policy for Problem 1. To this aim, we introduce the notion of violating transitions at a given set of markings. Note that given a marking m and a transition $t\in En(m)$, m_t denotes the marking reached by firing t at m.

Definition 1: Given a PN N, a GMEC (ω, k) and a set of markings $\mathcal{M}$, the set of *violating transitions* at $\mathcal{M}$ is

$$\Gamma_{(\omega, k)}(\mathcal{M})=\{t\in En(\mathcal{M}) \mid \exists m\in \mathcal{M}, \text{ s.t. } \omega m_t >k\}.$$

In words, $\Gamma_{(\omega, k)}(\mathcal{M})$ includes all those transitions whose firing at some marking in $\mathcal{M}$ leads to a marking violating the GMEC (ω, k).

Now, we compute an on-line control policy by Method 1, where δ denotes the current observation and $\mathcal{M}$ contains all possible markings consistent with δ. When nothing is observed, δ is initialized at ε that denotes the empty sequence and $\mathcal{M}$ is initialized at $\{m_0\}$. Accordingly, the disabled set $\rho(\varepsilon)$ is computed and enforced on the plant, which is exactly the set of violating transitions at $\{m_0\}$, namely, $\Gamma_{(\omega, k)}(\{m_0\})$. Then, every time a new transition t is observed, we update the set $\mathcal{M}$ of markings consistent with the observation and compute the new set $\Gamma_{(\omega,k)}(\mathcal{M})$ of violating transitions. In particular, the updating of $\mathcal{M}$ is done in Step 4, where the new set $\mathcal{M}$ contains the markings reached from a marking in the old set $\mathcal{M}$ by firing a transition that possibly produces the observation t and is not a disabled transition.

Method 1: On-line computation of an optimal policy

Input: 1) plant: a PN system (N, m_0) vulnerable to a replacement attack $\mathcal{A}$; and 2) specification: a GMEC (ω, k).

Output: A control policy ρ.

```
1. Initialization: $\delta\leftarrow\varepsilon$; $\mathcal{M}\leftarrow\{m_0\}$;
2. $\rho(\delta)\leftarrow\Gamma_{(\omega, k)}(\mathcal{M})$;
3. **while** there is an observation $t\in T$ **do**
4.     $\mathcal{M}\leftarrow\{m_{t'} \mid t'$ is enabled at $m\in \mathcal{M}, t'\in A^{-1}(t)\backslash\rho(\delta)\}$;
5.     $\delta\leftarrow\delta t$;
6.     $\rho(\delta)\leftarrow\Gamma_{(\omega, k)}(\mathcal{M})$;
7. **end while**
```

Example 2: Consider the PN system (N, m_0) in Fig. 2 with specification (ω, k): $m(p_2)\leq 1$, and assume that the system is vulnerable to a replacement attack $\mathcal{A}=\{(t_2, t_3)\}$. Method 1 computes the control policy as follows.

1) Nothing is observed, i.e., $\delta=\varepsilon$. It is $\mathcal{M}=\{m_0\}=\{[2, 0, 0]^T\}$ and thus $\rho(\varepsilon)=\Gamma_{(\omega, k)}(\mathcal{M})=\varnothing$;

2) Suppose that t_3 is observed. $\mathcal{M}$ is updated to $\{[1, 1, 0]^T, [1, 0, 1]^T\}$ since $A^{-1}(t_3)=\{t_2, t_3\}$ and $\rho(\varepsilon)=\varnothing$. Accordingly, it is $\rho(t_3)=\Gamma_{(\omega, k)}(\mathcal{M})=\{t_2\}$;

3) Suppose that t_3 is observed again. It is $A^{-1}(t_3)\backslash\rho(t_3)=\{t_3\}$.

Hence, $\mathcal{M}$ is updated to $\{[0,\ 1,\ 1]^T,\ [0,\ 0,\ 2]^T\}$ and $\rho(t_3t_3)=\Gamma_{(\omega,k)}(\mathcal{M})=\{t_4\}$.

Similarly, every time a new transition is observed, the disabled set is recomputed.

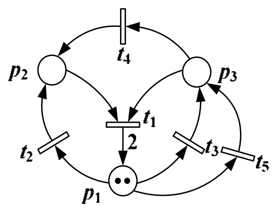

Fig. 2. PN system (N, m_0) in Examples 2-5

Theorem 1: Let (N, m_0) be a PN system vulnerable to a replacement attack $\mathcal{A}$ and (ω, k) be the imposed GMEC. The policy computed by Method 1 is optimal.

Proof: Let ρ be the policy computed by Method 1. It is trivial to see that ρ is acceptable, i.e., $R(N, m_0)|_\rho\subseteq\mathcal{L}_{(\omega,k)}$. Let ρ' be a policy more permissive than ρ. It means that there exists an observed sequence $\delta\in T^*$ generated by the plant such that 1) $\rho'(\delta')=\rho(\delta')$, $\forall\delta'\in\mathcal{P}_r(\delta)\setminus\{\delta\}$; and 2) $\rho'(\delta)\subset\rho(\delta)$. Let $\mathcal{M}$ be the set of markings consistent with the observation obtained before the last transition in δ is observed. Clearly, it is $\mathcal{M}\subseteq\mathcal{L}_{(\omega,k)}$. Since $\rho'(\delta)\subset\rho(\delta)$, there exists a violating transition associated with $\mathcal{M}$ that is not included in $\rho'(\delta)$. Hence, the system possibly reaches an illegal marking by firing such a transition. In other words, $R(N, m_0)|_{\rho'}\not\subseteq\mathcal{L}_{(\omega,k)}$. This proves that ρ is optimal. ■

Remark 1: Method 1 computes a control action every time the firing of a transition is observed. Thus, as long as the considered net system may keep running, the computation of control actions is always needed and will not terminate. This is why we describe the proposed control policy as "method" rather than "algorithm" since an algorithm should be able to terminate. Besides, we notice that Method 1 is not limited to GMECs. It is actually applicable to any control specification that requires a system to evolve within a marking set. ■

Remark 2: We analyze the computational complexity of Method 1. In Method 1, we record a set $\mathcal{M}$ of markings consistent with an observation. At the beginning, it contains the initial marking only. When we observe the first transition, we update the set $\mathcal{M}$. Since, in the worst case, all the transitions in the net may have fired, the updated set $\mathcal{M}$ contains at most $|T|$ markings. Then, we compute $\Gamma_{(\omega,k)}(\mathcal{M})$ that is the disabled set relative to the current observation. By its definition, we should check for each marking in $\mathcal{M}$ and each enabled transition. Thus, the complexity of computing such a control action is $O(|T|^2)$. When we observe the second transition, we update again the set $\mathcal{M}$. Similarly, considering the worst case, the updated set $\mathcal{M}$ contains at most $|T|^2$ markings. Then, we compute again $\Gamma_{(\omega,k)}(\mathcal{M})$ and its complexity is $O(|T|^3)$. By repeating the above reasoning, when we observe the nth transition, the updated set $\mathcal{M}$ contains at most $|T|^n$ markings and the complexity of computing a control action is $O(|T|^{n+1})$. We notice that the set $\mathcal{M}$ is always bounded although its size grows exponentially with the length of the observed sequence. Nevertheless, since the size of $\mathcal{M}$ grows exponentially, the computation of a control action might become inefficient particularly in large-size systems. Consequently, we look for control policies with a smaller computational complexity. ■

V. Control Policies Based on a Monitor-controlled PN System

In this section, we develop two control policies based on the off-line construction of a *monitor-controlled PN system*, which reveal more efficient than Method 1.

A. Monitor-controlled PN System

Given a plant (N, m_0) and a GMEC (ω, k), the *monitor-controlled PN system*, denoted as (N^c, m_0^c), is an augmented system obtained by adding, to the net system to be controlled, monitor p_c that enforces GMEC (ω, k) via the place-invariant method [16]. In more detail, it is

$$[N^c]=\begin{bmatrix}[N]\\ -\omega\cdot[N]\end{bmatrix}=\begin{bmatrix}[N]\\ -\varpi\end{bmatrix}\ \text{and}\ m_0^c=\begin{bmatrix}m_0\\ k-\omega\cdot m_0\end{bmatrix},$$

where $[N^c]$ and $[N]$ are the incidence matrices of N^c and N, respectively. Note that the last rows of $[N^c]$ and m_0^c correspond to monitor p_c.

Example 3: Consider the PN system (N, m_0) in Fig. 2 and the GMEC (ω, k): $m(p_2)\leq 1$. The corresponding monitor-controlled PN system (N^c, m_0^c) is shown in Fig. 3.

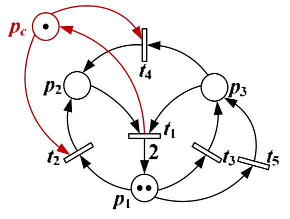

Fig. 3. Monitor-controlled PN system (N^c, m_0^c) relative to the PN system in Fig. 2 and the GMEC (ω, k): $m(p_2)\leq 1$

In the following, we use m^c to indicate the marking of the monitor-controlled PN N^c. By default, m^c is divided into two parts, namely,

$$m^c=[m^{\mathrm{T}}\quad m_{pc}^{\mathrm{T}}]^{\mathrm{T}},$$

where m denotes the marking corresponding to the original PN N and m_{pc} denotes the marking of the monitor place p_c.

We now introduce some notation to be used in the rest of the paper.

Definition 2: Given a monitor-controlled PN $N^c=(P^c, T^c, F^c, W^c)$ and a marking m_{pc} of the monitor place p_c, we define the set of *monitor-disabled transitions* as

$$Dis(m_{pc})=\{t\in p_c^\bullet \mid m_{pc}<W^c(p_c, t)\}.$$

Definition 3: Given a monitor-controlled PN $N^c=(P^c, T^c, F^c, W^c)$ and a marking $m^c=[m^T\ m_{pc}^T]^T$, we define:

$$\Psi(m^c)=Dis(m_{pc})\cap En(m).$$

In other words, $\Psi(m^c)$ is the set of transitions that are enabled in the original net system but are disabled by monitor p_c. Furthermore, given a set of markings $\mathcal{M}^c$ of N^c, we define:

$$\Psi(\mathcal{M}^c)=\bigcup_{m^c\in\mathcal{M}^c}\Psi(m^c).$$

Example 4: Consider the monitor-controlled PN N^c in Fig. 4

at marking m^c=[1, 1, 0, 0]^T. It is m=[1, 1, 0]^T and m_{pc}=0. Hence, $Dis(m_{pc})=\{t_2, t_4\}$ and $\Psi(m^c)=\{t_2\}$ by Definitions 2 and 3.

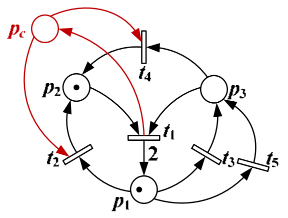

Fig. 4. Monitor-controlled PN N^c at marking m^c=[1, 1, 0, 1]^T

B. Optimal control policy

In this subsection, we propose the second approach called Method 2, to derive a control policy. Method 2 is similar to Method 1. The main difference is that Method 2 constructs a monitor-controlled PN system (N^c, m_0^c) off-line and the policy is computed on-line based on (N^c, m_0^c). Every time a transition is observed, Method 2 computes the set $\mathcal{M}^c$ of all possible reachable markings of (N^c, m_0^c) and determines the disabled set based on $\mathcal{M}^c$. Actually, the policy computed by Method 2 is the same as that computed by Method 1. It means that Method 2 also derives an optimal solution, which is formally presented and proven as follows.

```
Method 2: Computation of an optimal policy based on monitors
Input: 1) plant: a PN system ($N$, $m_0$) vulnerable to a replacement attack $\mathcal{A}$; and 2) specification: a GMEC ($\omega$, $k$).
Output: A control policy $\rho$.
Off-line computation:
1. Compute the monitor-controlled PN system ($N^c$, $m_0^c$) relative to the net ($N$, $m_0$) and the GMEC ($\omega$, $k$).
On-line computation and control:
1. Initialization: $\delta\leftarrow\varepsilon$; $\mathcal{M}^c\leftarrow\{m_0^c\}$;
2. $\rho(\delta)\leftarrow\Psi(\mathcal{M}^c)$ ;
3. while there is an observation $t\in T$ do
4.     $\mathcal{M}^c\leftarrow\{m^c_{t'} \mid t'$ is enabled at $m^c\in\mathcal{M}^c$, $t'\in A^{-1}(t)\backslash\rho(\delta)\}$;
5.     $\delta\leftarrow\delta t$;
6.     $\rho(\delta)\leftarrow\Psi(\mathcal{M}^c)$;
7. end while
```

The following results hold.

Property 1 [16]: Let (N, m_0) be a PN system, (ω, k) be a GMEC, and (N^c, m_0^c) be the corresponding monitor-controlled PN system. Given a marking $m^c\in R(N^c, m_0^c)$, it holds that

1) $\omega m+m_{pc}=k$; and

2) $Dis(m_{pc})=\{t\in T \mid \omega m+\overline{\omega}(t)>k\}$.

Property 2: Let (N, m_0) be a PN system, (ω, k) be a GMEC, and (N^c, m_0^c) be the corresponding monitor-controlled PN system.

1) Given a marking $m^c\in R(N^c, m_0^c)$, it holds that $\Psi(m^c)=\{t\in En(m) \mid \omega m_t >k\}$;

2) Given a set of markings $\mathcal{M}^c\subseteq R(N^c, m_0^c)$, it holds that $\Psi(\mathcal{M}^c)=\Gamma_{(\omega, k)}(\mathcal{M})$, where $\mathcal{M}$ is the set of markings derived by restricting markings in $\mathcal{M}^c$ to the net N.

Proof: Straightforward from Property 1 and Definitions 1 and 3. ■

Theorem 2: Let (N, m_0) be a PN system vulnerable to a replacement attack $\mathcal{A}$ and (ω, k) be a GMEC. The policy computed by Method 2 is optimal.

Proof: Let ρ_2 and ρ_1 be policies computed by Methods 2 and 1, respectively. It is trivial to see that $\rho_2=\rho_1$ by Property 2. As a result, ρ_2 is optimal since ρ_1 is optimal. ■

Example 5: Consider again the PN system (N, m_0) in Fig. 2 subject to GMEC (ω, k): $m(p_2)\leq 1$ and vulnerable to a replacement attack $\mathcal{A}=\{(t_2, t_3)\}$. Now, we use Method 2 to compute a control policy. First, a monitor-controlled PN system (N^c, m_0^c) is constructed off-line, as shown in Fig. 3. Next, the on-line computation and control is performed based on (N^c, m_0^c). We show a part of the procedure as follows.

1) Nothing is observed, i.e., $\delta=\varepsilon$. It is $\mathcal{M}^c=\{m_0^c\}=\{[2, 0, 0, 1]^T\}$ and thus $\rho(\varepsilon)=\Psi(\mathcal{M}^c)=\emptyset$.

2) Suppose that t_3 is observed. Since $A^{-1}(t_3)=\{t_2, t_3\}$ and $\rho(\varepsilon)=\emptyset$, either t_2 or t_3 may have fired, resulting in $\mathcal{M}^c=\{m_1^c, m_2^c\}=\{[1, 1, 0, 0]^T, [1, 0, 1, 1]^T\}$. For sake of clarity, markings m_1^c and m_2^c are both shown in Fig. 5(a) in blue and black color, respectively. We can see that $\rho(t_3)=\Psi(\mathcal{M}^c) =\Psi(m_1^c)\cup\Psi(m_2^c)=\{t_2\}\cup\emptyset=\{t_2\}$.

3) Suppose that t_3 is observed again. It is $A^{-1}(t_3)\backslash\rho(t_3)=\{t_3\}$, i.e., t_3 has fired for sure. Hence, $\mathcal{M}^c$ is updated at $\mathcal{M}^c=\{m_3^c, m_4^c\}=\{[0, 1, 1, 0]^T, [0, 0, 2, 1]^T\}$, which is shown in Fig. 5(b) with m_3^c in blue and m_4^c in black. We can see that $\rho(t_3t_3)=\Psi(\mathcal{M}^c)=\Psi(m_3^c)\cup\Psi(m_4^c)=\{t_4\}\cup\emptyset=\{t_4\}$.

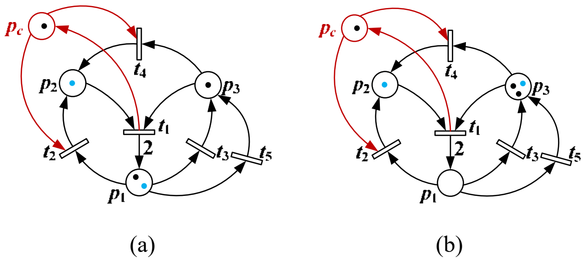

Fig. 5 (a) Marking set $\mathcal{M}^c=\{m_1^c, m_2^c\}$ consistent with $\delta=t_3$; and (b) marking set $\mathcal{M}^c=\{m_3^c, m_4^c\}$ consistent with $\delta=t_3t_3$.

Remark 3: Method 2 provides an alternative way to compute an optimal control policy. It consists of both off-line and on-line computations. Regarding the off-line computations, the complexity is proportional to the number of GMECs [16]. The on-line computations are similar to Method 1. The differences lie in that when implementing Method 2 we record the set $\mathcal{M}^c$ of markings of the monitor-controlled system consistent with an observation and based on that, we compute the disabling set $\Psi(\mathcal{M}^c)$. Thus, as in Method 1, when we observe the nth transition, the updated set $\mathcal{M}^c$ contains no more than $|T|^n$ markings. Then, the complexity of computing $\Psi(\mathcal{M}^c)$ is $O(|T|^n)$. Clearly, compared with Method 1, computing a control action using Method 2 is faster. However, the applicability of Method 2 could still become prohibitive for long evolutions of the system. ■

C. Non-optimal control policy with improved computational efficiency

In this subsection, we present a more efficient method (Method 3) to compute a control policy, which is still based on constructing the monitor-controlled PN system. The derived policy is acceptable but not necessarily optimal.

Method 3: Computation of a non-optimal policy based on monitors

Input: 1) plant: a PN system (N, m_0) vulnerable to a replacement attack $\mathcal{A}$; and 2) specification: a GMEC (ω, k).

Output: A control policy ρ.

```
Off-line computation:
1. Compute the monitor-controlled PN system ($N^c$, $m_0^c$) relative to the net ($N$, $m_0$) and the GMEC ($\omega$, $k$).
On-line computation and control:
1.  Initialization: $\delta \leftarrow \varepsilon$; $m^c \leftarrow m_0^c$; Flag←True;
2.  $\rho(\delta) \leftarrow Dis(m_{pc})$;
3.  while there is an observation $t \in T$ do
4.      if Flag=True then
5.          $T_{real} \leftarrow A^{-1}(t) \cap En(m^c)$;
6.          if $T_{real}$ is a singleton and $t' \in T_{real}$ then
7.              $m^c \leftarrow m^c + [N^c](\cdot, t')$;
8.          else
9.              $m_{pc} \leftarrow m_{pc} + \min_{t' \in T_{real}} [N^c](p_c, t')$;
10.             Flag←False;
11.         end if
12.     else
13.         $T_{real} \leftarrow A^{-1}(t) \setminus \rho(\delta)$;
14.         $m_{pc} \leftarrow m_{pc} + \min_{t' \in T_{real}} [N^c](p_c, t')$;
15.     end if
16.     $\delta \leftarrow \delta t$;
17.     $\rho(\delta) \leftarrow Dis(m_{pc})$;
18. end while
```

We explain Method 3 as follows. As in Method 2, we construct off-line the monitor-controlled PN system (N^c, m_0^c) and then perform the on-line computation and control based on it.

We introduce a variable Flag∈{True, False}. In particular, Flag=True indicates that we know with certainty the marking of the considered system before making a new observation, while Flag=False means the opposite, i.e., we do not know with certainty in which marking the system is when a new observation is done. Hence, Flag is initialized at "True" since we know that the system is in marking m_0^c when we start our observation. Besides, m^c is initialized at m_0^c, representing the system marking and the disabled set $\rho(\varepsilon)$ enforced on the plant is the set of monitor-disabled transitions computed based on the initial token-count of p_c. Whenever a new observation is produced, the token-count of monitor p_c is updated and the disabled set enforced on the plant is updated accordingly based on the token-count of p_c. In what follows, we focus on the way the token-count of p_c is updated when a new observation is produced. In particular, two different cases are considered:

Case 1: We know with certainty in which marking the system is before such an observation (i.e., Flag=True); and

Case 2: We do not know with certainty in which marking the system is before such an observation (i.e., Flag=False).

For the former case, we compute the set of transitions that may have fired (stored in T_{real}) based on the known current marking and the observed transition as in Step 5. Clearly, if T_{real} contains only one transition, the new marking can be uniquely determined including the token-count in p_c. Otherwise, as it typically occurs, we do not know the new marking of the system. In this case, Method 3 does not enumerate possible new markings of the system but only updates the token-count in p_c considering the firing effect of each transition in T_{real} on p_c. More precisely, the token-count in p_c is updated considering the firing of a transition in T_{real} such that p_c contains the fewest tokens. In such a case, Flag is updated to "False" and remains like that until the end of the system observation.

When Flag=False, i.e., the current marking of the system is not known, we do not reconstruct the marking of the system but update the token-count in p_c only. In this case, we compute the set of transitions that may have fired only based on the observed transition and the enforced disabled set as in Step 13. Then, we update again the token-count in p_c considering the firing of a transition in T_{real} such that p_c contains the fewest tokens.

Theorem 3: Let (N, m_0) be a PN system vulnerable to a replacement attack $\mathcal{A}$ and (ω, k) be a GMEC. The policy computed by Method 3 is acceptable.

Proof: See the supplementary material in [21]. ■

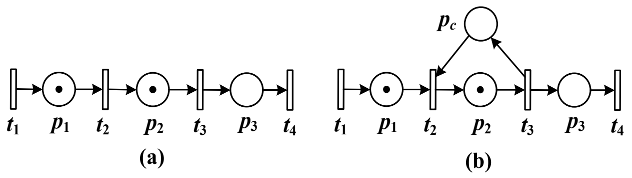

Fig. 6 (a) Plant: a PN system (N, m_0); and (b) Monitor-controlled PN system (N^c, m_0^c) associated with GMEC (ω, k): $m(p_2) \leq 1$

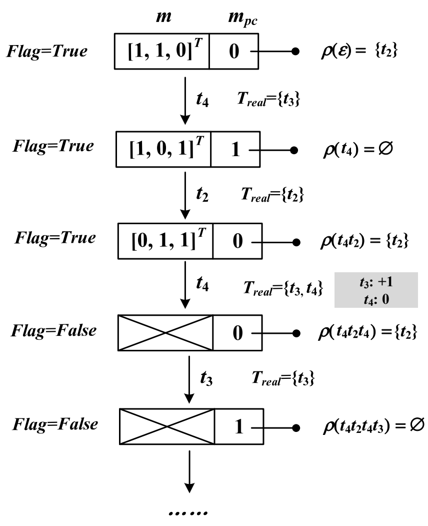

Fig. 7 Illustration of the on-line control in Example 6

Example 6: Consider the PN system (N, m_0) in Fig. 6(a) subject to a GMEC (ω, k): $m(p_2) \leq 1$ and vulnerable to a replacement attack $\mathcal{A}=\{(t_3, t_4)\}$. Let us see how Method 3 works. First, the monitor-controlled PN system (N^c, m_0^c) is constructed off-line, as shown in Fig. 6(b). Next, the on-line computation and control is performed based on (N^c, m_0^c). We explain a part of the procedure as follows, which is also shown in Fig. 7 for sake of clarity.

1) Nothing is observed, i.e., $\delta=\varepsilon$. We initialize Flag=True and $m^c=m_0^c=[1, 1, 0, 0]^T$. Note that the last entry of m^c indicates the token-count of p_c. Thus, $\rho(\varepsilon)=Dis(m_{pc})=\{t_2\}$.
2) Suppose that t_4 is observed. Since Flag=True, it is $T_{real}=A^{-1}(t_4)\cap En(m^c)=\{t_3\}$. Hence, m^c is updated by firing t_3 instead of t_4, resulting in $m^c=[1, 0, 1, 1]^T$. Hence, $\rho(t_4)=Dis(m_{pc})=\varnothing$.
3) Suppose that t_2 is observed. Since Flag=True, $T_{real}=A^{-1}(t_2)\cap En(m^c)=\{t_2\}$. Hence, m^c is updated at $m^c=[0, 1, 1, 0]^T$ by firing t_2. Thus, $\rho(t_4t_2)=Dis(m_{pc})=\{t_2\}$.
4) Suppose that t_4 is observed. Since Flag is still "True", it is $T_{real}=A^{-1}(t_4)\cap En(m^c)=\{t_3, t_4\}$. Now, $|T_{real}|=2$. Thus, we only update the token-count of p_c. Since $[N^c](p_c, t_3)=1$ and $[N^c](p_c, t_4)=0$, it follows that $m_{pc}=m_{pc}+\min\{[N^c](p_c, t_3), [N^c](p_c, t_4)\}=0$. Hence, $\rho(t_4t_2t_4)=Dis(m_{pc})=\{t_2\}$. Finally, Flag is updated to False.
5) Suppose that t_3 is observed. Since Flag=False, it is $T_{real}=A^{-1}(t_3)\backslash\rho(t_4t_2t_4)=\{t_3\}$. Hence, $m_{pc}= m_{pc}+[N^c](p_c, t_3)=1$. Accordingly, $\rho(t_4t_2t_4t_3)=Dis(m_{pc})=\varnothing$.

Example 7: Consider again the problem in Example 5. Now, we use Method 3 to compute a control policy. The monitor-controlled PN system (N^c, m_0^c) is still the system in Fig. 3. Let us see a part of the on-line computation procedure.

1) Nothing is observed, i.e., $\delta=\varepsilon$. It is Flag=True and $m^c=m_0^c=[2, 0, 0, 1]^T$. Thus, $\rho(\varepsilon)=Dis(m_{pc})=\varnothing$;

2) Suppose that t_3 is observed. Since Flag=True, it is $T_{real}=A^{-1}(t_3)\cap En(m^c)=\{t_2, t_3\}$. Hence, we update m_{pc} such that $m_{pc}=m_{pc}+\min\{[N^c](p_c, t_2), [N^c](p_c, t_3)\}=1+\min\{-1, 0\}=0$. Then, $\rho(t_3)=Dis(m_{pc})=\{t_2, t_4\}$ and Flag is updated to False.

3) Suppose that t_3 is observed again. Since Flag=False, $T_{real}=A^{-1}(t_3)\backslash\rho(t_3)=\{t_3\}$. Hence, m_{pc} is updated at $m_{pc}= m_{pc}+[N^c](p_c, t_3)=0$. Accordingly, $\rho(t_3t_3)=Dis(m_{pc})=\{t_2, t_4\}$.

We observe that compared with the policy computed by Method 2, which is shown in Example 5, the policy computed by Method 3 is more restrictive.

Remark 4: The non-optimality of Method 3 can be explained as follows. Method 3 does not enumerate the set of markings in which the system can be when such a set is not a singleton. Indeed, in such a case, it only updates the token-count in monitor p_c to make it restrictive enough to prevent the system from entering an illegal marking. Since the enumeration of markings is avoided, the computation is much more efficient. However, it results in a control policy not necessarily optimal. In particular, the set T_{real} of possible really fired transitions computed at Flag=False may include non-enabled transitions since we do not have the information on the marking of the system. As a result, the set of transitions disabled by the computed monitor p_c can be larger than it needs to be, which turns out a more restrictive control policy. Such an effect could increase when the evolution of the system proceeds. ■

Remark 5: As Method 2, the complexity of the off-line computations in Method 3 is proportional to the number of GMECs [16]. Regarding the on-line computations of Method 3, the complexity of computing a control action is always $O(|T|)$ since we always record one marking only and every time we observe a transition, no more than $|T|$ transitions should be enumerated. As a result, Method 3 is also applicable to large-size nets. ■

In what follows, we show that the optimality of the policy computed by Method 3 is guaranteed when a certain condition on the replacement attack is satisfied. The condition can be interpreted as that the variation of the token-count in the monitor p_c is identical for all transitions that could have fired corresponding to an observed transition under the replacement attack.

Theorem 4: Let (N, m_0) be a PN system vulnerable to a replacement attack $\mathcal{A}$ and (ω, k) be a GMEC. The policy computed by Method 3 is optimal if $\forall t\in T$, it holds that $\varpi(t_1)=\varpi(t_2)=\ldots=\varpi(t_n)$, where $\{t_1, t_2, \ldots, t_n\}=A^{-1}(t)$.

Proof: See the supplementary material in [21]. ■

If the condition in Theorem 4 is satisfied, Method 3 can be simplified by reducing Steps 4-15 to the following two steps:

1. wait for an observation $t\in T$ from the plant;
2. $m_{pc}\leftarrow m_{pc}+ [N^c](p_c,t)$; /*Equivalently, $m_{pc}\leftarrow m_{pc}-\varpi(t)$;*/

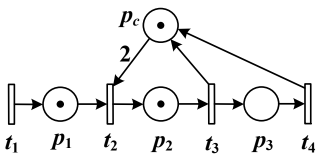

Fig. 8 Monitor-controlled PN system $(N^{c\prime}, m_0^{c\prime})$ associated with the plant (N, m_0) in Fig. 6 (a) and the GMEC (ω', k'): $2m(p_2)+m(p_3)\leq 3$

Remark 6: Inspired by Theorem 4, we have another idea to compute a control policy with timely response. That is, we may transform the given GMEC into a more restrictive GMEC that satisfies the condition in Theorem 4 and then perform the simplified Method 3 to compute a control policy. Clearly, the optimality is sacrificed to enhance the computational efficiency. For instance, GMEC (ω, k): $m(p_2)\leq 1$ in Example 6 can be transformed into a more restrictive GMEC (ω', k'): $2m(p_2)+m(p_3)\leq 3$ such that $\varpi'(t_3)=\varpi'(t_4)=-1$. The corresponding monitor-controlled PN system $(N^{c\prime}, m_0^{c\prime})$ is shown in Fig. 8. We thus can compute an on-line efficient control policy by the simplified Method 3, which is optimal for GMEC (ω', k'): $2m(p_2)+m(p_3)\leq 3$ but only acceptable for the original GMEC (ω, k): $m(p_2)\leq 1$. Normally, we want to get a transformed GMEC as permissive as possible. How to do systematically such GMEC transformation is left as future work. ■

We finally note that the proposed methods can be easily extended to a control specification that is a conjunction of multiple GMECs. Suppose that a set of GMECs is $\Omega=\{(\omega_1, k_1), (\omega_2, k_2), \ldots, (\omega_i, k_i)\}$. Let us see how the proposed methods are modified to handle the specification $\wedge\Omega$.

Consider Method 1. In this case, we only need to extend the set of violating transitions $\Gamma_{(\omega, k)}(\mathcal{M})$ to Ω such that

$$\Gamma_\Omega(\mathcal{M})=\{t\in En(\mathcal{M}) \mid \exists m\in \mathcal{M}, \exists(\omega, k)\in\Omega, \text{ s.t. } \omega m_t >k\}.$$

Consider Method 2. The monitor-controlled PN system (N^c, m_0^c) is now an augmented system by adding monitors $p_{c1}, p_{c2}, \ldots, p_{ci}$ to (N, m_0), each monitor enforcing a GMEC in Ω by the place-invariant method. Specifically, it is

$$[N^c]=[[N],\quad -\varpi_1,\quad -\varpi_2,\quad \ldots,\quad -\varpi_i]^{\mathrm{T}};$$

$$m_0^c=[m_0,\quad k-\omega_1\cdot m_0,\quad k-\omega_2\cdot m_0,\quad \ldots,\quad k-\omega_i\cdot m_0]^{\mathrm{T}},$$

where the last i rows of $[N^c]$ and m_0^c correspond to monitors p_{c1}, p_{c2}, …, and p_{ci}, respectively. We use P_c to denote the set of monitors, i.e., $P_c=\{p_{c1}, p_{c2}, \ldots, p_{ci}\}$. Thus, given a marking m^c of the monitor-controlled PN N^c, it can be divided into two parts, that is, $m^c=[m \quad m_{Pc}]^T$, where m and m_{Pc} denote markings by restricting m^c within N and P_c, respectively. In addition, the set of monitor-disabled transitions is extended to marking m_{Pc} of P_c such that $Dis(m_{Pc})=\bigcup_{pc\in Pc} Dis(m_{pc})$. Accordingly, given a marking m^c of a monitor-controlled PN N^c, $\Psi(m^c)$ is re-defined as $\Psi(m^c)=Dis(m_{Pc})\cap En(m)$.

It is trivial to see that Methods 1 and 2, appropriately modified as mentioned above, both result in an optimal control policy when handling a conjunction of GMECs.

Consider Method 3. In addition to computing a monitor-controlled PN system (N^c, m_0^c) w.r.t. the conjunction of GMECs $\wedge\Omega$, Steps 9 and 14 are replaced by the step:

$$\forall p_c\in P_c,\ m_{pc}\leftarrow m_{pc}+\min_{t'\in T_{real}}[N^c](p_c,t')$$

and Steps 2 and 17 are replaced by $\rho(\delta)\leftarrow Dis(m_{Pc})$. In this way, the modified Method 3 leads to an acceptable control policy with timely response. Furthermore, it is optimal if $\forall t\in T$, $\forall(\omega, k)\in\Omega$, $\varpi(t_1)=\varpi(t_2)=\ldots=\varpi(t_n)$, where $\{t_1, t_2, \ldots, t_n\}=A^{-1}(t)$.

We finally notice that, a simple case study is presented in [21], which is not reported here for space limitations.

VI. Conclusions and Future Work

This paper focuses on the problem of designing a supervisory control law in Petri net (PN) systems under attack. We consider the control specifications given in terms of Generalized Mutual Exclusion Constraints and deal with replacement attacks that may disguise the occurrence of certain events as the occurrence of other events. Specifically, three methods are proposed to obtain on-line control policies. Method 1 derives an optimal policy requiring marking enumeration; so it is limited by the size of the considered system. Method 2 also provides an optimal policy. It introduces monitors to the plant by a place-invariant method but still requires marking enumeration and thus suffers from similar problems as Method 1. To improve the on-line computational efficiency, we propose Method 3 that computes a control policy by avoiding marking enumeration. Consequently, Method 3 ensures a timely response of the computed policy but at the expense of its optimality. The major contribution of the paper consists in being the first paper dealing with supervisory control under attack in the framework of PNs. In particular, the proposed methods may handle unbounded PN systems, whereas methods in the literature are all based on finite-state automata and thus limited to bounded systems.

In our future work, we intend to study more general control problems under attack. We may formulate the problem in the formalism of PNs with non-deterministic observations, which allows us to consider more general attacks that are able to replace and hide observations. In addition, we may take into account uncontrollable and/or unobservable transitions. Finally, we may consider other types of attacks.

References

[1] M. Agarwal, S. Biswas, and S. Nandi, "Discrete event system framework for fault diagnosis with measurement inconsistency: Case study of rogue dhcp attack," *IEEE/CAA Journal of Automatica Sinica,* vol. 6, no. 3, pp. 789-806, 2019.

[2] M. Agarwal, S. Purwar, S. Biswas, and S. Nandi, "Intrusion detection system for ps-poll dos attack in 802.11 networks using real time discrete event system," *IEEE/CAA Journal of Automatica Sinica,* vol. 4, no. 4, pp. 792-808, 2017.

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

[4] A. Giua, F. DiCesare, and M. Silva, "Generalized mutual exclusion contraints on nets with uncontrollable transitions," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, Chicago, IL, USA, 1992: IEEE, pp. 974-979.

[5] L. E. Holloway, B. H. Krogh, and A. Giua, "A survey of petri net methods for controlled discrete event systems," *Discrete Event Dynamic Systems,* vol. 7, no. 2, pp. 151-190, 1997.

[6] P. M. Lima, "Security against network attacks in supervisory control systems," Master's thesis, Federal University of Rio de Janeiro, 2017.

[7] P. M. Lima, M. V. S. Alves, L. K. Carvalho, and M. V. Moreira, "Security against communication network attacks of cyber-physical systems," *Journal of Control, Automation and Electrical Systems,* vol. 30, no. 1, pp. 125-135, 2019.

[8] P. M. Lima, L. K. Carvalho, and M. V. Moreira, "Detectable and undetectable network attack security of cyber-physical systems," *IFAC-PapersOnLine,* vol. 51, no. 7, pp. 179-185, 2018.

[9] F. Lu, Q. Zeng, M. Zhou, Y. Bao, and H. Duan, "Complex reachability trees and their application to deadlock detection for unbounded petri nets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 49, no. 6, pp. 1164-1174, 2019.

[10] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE,* vol. 77, no. 4, pp. 541-580, 1989.

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

[12] D. Thorsley and D. Teneketzis, "Intrusion detection in controlled discrete event systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006: IEEE, pp. 6047-6054.

[13] T. Ushio and S. Takai, "Nonblocking supervisory control of discrete event systems modeled by mealy automata with nondeterministic output functions," *IEEE Transactions on Automatic Control,* vol. 61, no. 3, pp. 799-804, 2016.

[14] M. Wakaiki, P. Tabuada, and J. P. Hespanha, "Supervisory control of discrete-event systems under attacks," *arXiv preprint arXiv:1701.00881,* 2017.

[15] S. Xu and R. Kumar, "Discrete event control under nondeterministic partial observation," in *IEEE International Conference on Automation Science and Engineering*, 2009: IEEE, pp. 127-132.

[16] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis, "Feedback control of petri nets based on place invariants," *Automatica,* vol. 32, no. 1, pp. 15-28, 1996.

[17] X. Yin, "Supervisor synthesis for mealy automata with output functions: A model transformation approach," *IEEE Transactions on Automatic Control,* vol. 62, no. 5, pp. 2576-2581, 2017.

[18] J. Zaytoon and S. Lafortune, "Overview of fault diagnosis methods for discrete event systems," *Annual Reviews in Control,* vol. 37, no. 2, pp. 308-320, 2013.

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

[20] Q. Zhang, C. Seatzu, A. Giua, Z. Li, "Stealthy sensor attacks for plants modeled by labeled Petri nets," 15th IFAC Workshop on Discrete Event Systems, Rio de Janeiro, Brazil, November 11-13, 2020 (to appear).

[21] [Online]. Available: https://corsi.unica.it/ingegneriaelettricaeelettronica/files/2020/12/Supplementary-12.18.pdf