Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 53-4 (2020) 14–20

Stealthy Sensor Attacks for Plants Modeled by Labeled Petri Nets [⋆]

Qi Zhang [*] Carla Seatzu [**] Zhiwu Li [*] Alessandro Giua [**]

[*] *SEME, Xidian University, Xi'an 710071, China (e-mail: qzhang_3@stu.xidian.edu.cn, zhwli@xidian.edu.cn)*

[**] *DIEE, University of Cagliari, 09123 Cagliari, Italy (e-mail: {seatzu, giua}@diee.unica.it)*

Abstract: The problem of stealthy sensor attacks for labeled Petri nets is considered. An operator observes the plant to establish if a set of critical markings has been reached. The attacker can corrupt the sensor channels that transmit the sensor readings, making the operator incapable to establish when a critical marking is reached. We first construct the stealthy attack Petri net that keeps into account the real plant evolutions observed by the attacker and the corrupted plant evolutions observed by the operator. Starting from the reachability graph of the stealthy attack Petri net, an attack structure is defined: it describes all possible attacks. The supremal stealthy attack substructure can be obtained by appropriately trimming the attack structure. An attack function is effective if the supremal stealthy attack substructure contains a state whose first element is a critical marking and the second element is a noncritical marking.

Keywords: Discrete event systems, labeled Petri nets, cyber-physical systems, sensor attacks.

1. INTRODUCTION

The problem of cyber attacks in continuous systems has been recently considered by Kim et al. (2019) and Zhang et al. (2019a). In the framework of discrete event systems, two kinds of attacks are considered: sensor attacks (Su, 2018; Wakaiki et al., 2018), and actuator enablement attacks (Zhu et al., 2019; Lin et al., 2019). The attacker can corrupt the sensor channels between the plant and the supervisor, inserting fake sensor readings in the supervisor observation or erasing some sensor readings from the supervisor observation, making the closed-loop system reach unsafe states. The attacker can also corrupt the control commands from the supervisor to the plant, enabling events that are disabled by the supervisor, again leading the closed-loop system to unsafe states.

Several authors consider both sensor and actuator enablement attacks (Carvalho et al., 2018; Lima et al., 2019). Wang et al. (2019) study the problem of supervisory control of discrete event systems under sensor and actuator enablement attacks. They model the supervisor as a *finite state transducer*, which allows the supervisor to counter the attacker by modifying the sensor readings received by the attacker. New controllability conditions are proposed and algorithms to synthesize resilient supervisors are p-resented. Góes et al. (2019) investigate the problem of sensor attacks for a system modeled as a *probabilistic finite state automaton*, developing an optimal attack strategy that allows the supervised system under attack to have the maximal likelihood to reach the unsafe states.

This paper differs from most of the existing works (Carvalho et al., 2018; Su, 2018) that investigate the problem of cyer attack at the supervisory control layer. Indeed, we study such a problem at the observation layer. In addition, we formulate the problem using *labeled Petri nets* (LPNs) as the reference model. In more details, the evolution of the plant is observed by an operator, and the goal of the operator is to establish if a set of *critical markings* is reached, in which case appropriate protection actions for the plant are activated. We assume that the attacker has a complete knowledge of the plant model, and it may insert in the operator's observation fake labels or erase labels that have generated by the plant with the objective of preventing the operator to realize when the system reaches some critical marking. Furthermore, the attacker wants to remain stealthy, namely the operator should not be able to detect the presence of the attacker.

We believe that modelling an attack problem with Petri nets may lead to the use of more efficient approaches for its analysis. Well known examples are structural approaches–e.g., state equation or *generalized mutual exclusion constraint* (GMEC) formalisms–which have been successfully used for supervisory control and deadlock analysis (Giua et al., 1992; Ma et al., 2016). Other efficient analysis techniques are based on semi-structural approaches–e.g., those based on *basis markings*–that alleviate the need for enumerating the full reachability space and have been used for diagnosis and opacity (Cabasino et al., 2011; Tong et al., 2017). However, the results presented here are only a first preliminary step in this general direction: we focus on how a Petri net model for cyber attacks can be

[⋆] This work is partially supported by the National Key R&D Program of China under Grant 2018YFB1700104, the Natural Science Foundation of China under Grand Nos. 61472295, 61673309, 61873342, the ShaanXi Huashan Scholars, the Science and Technology Development Fund, MSAR, under Grant No. 122/2017/A3. This work is also partially supported by Project RASSR05871 MOSIMA funded by Region Sardinia, FSC 2014-2020, Annualita' 2017, Area 3, Action Line 3.1.

10.1016/j.ifacol.2021.04.048

constructed but only use its reachability graph for analysis. The possibility of using more efficient approaches will be addressed in future work.

2. PRELIMINARIES

2.1 Petri Nets

A *Petri net* is a four-tuple $N = (P, T, Pre, Post)$, where P is a set of m *places* represented by circles, T is a set of n *transitions* represented by bars, $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the *pre−* and *post−incidence functions* that specify the arcs directed from places to transitions and from transitions to places, respectively, where $\mathbb{N}$ is the set of non-negative integers. The *incidence matrix* of a net is defined by $C = Post - Pre$.

A *marking* is a vector $M : P \to \mathbb{N}$ that assigns to each place of the net a non-negative integer, represented by black dots. The marking of a place P is denoted as $M(p)$. A marking is also denoted by $M = \sum_{p\in P} M(p)\cdot p$. A transition t is *enabled* at marking M if $M \geq Pre(\cdot, t)$, and $M' = M + C(\cdot, t)$ if t fires. We use $M[\sigma\rangle$ to denote a sequence $\sigma \in T^*$ is enabled at marking M, and $M[\sigma\rangle M'$ to denote the firing of σ yielding marking M'.

We use $\langle N, M_0\rangle$ to denote a net N with initial marking M_0. The *reachability set* denoted as $R(N, M_0)$, is the set of all markings that are reachable from the initial marking M_0.

2.2 Labeled Petri Nets

A *labeled Petri net* (LPN) is a 4-tuple $G = (N, M_0, E, \ell)$, where $\langle N, M_0\rangle$ is a Petri net structure, E is the *alphabet* (a set of labels), and $\ell : T \to E \cup \{\varepsilon\}$ is the *labeling function* that assigns to each transition $t \in T$ either a label from E or the empty word ε.

The set of transitions $T = T_o \cup T_{uo}$, where T_o is the set of *observable* transitions, and T_{uo} is the set of *unobservable* transitions. The labeling function can be extended to a sequence of transitions $\ell : T^* \to E^*$, i.e., $\ell(\sigma t) = \ell(\sigma)\ell(t)$, where $\sigma \in T^*$, and $t \in T$. The *inverse labeling function* $\ell^{-1} : E^* \to 2^{T^*}$ is defined as $\ell^{-1}(s) = \{\sigma \in T^* \mid \ell(\sigma) = s\}$, where $s \in E^*$.

We use $L(N, M_0) = \{\sigma \in T^* \mid (\exists M \in R(N, M_0))\ M_0[\sigma\rangle M\}$ to denote the set of all sequences of transitions that can be fired at the initial marking M_0. The *generated language* of G is defined as $\mathcal{L}(G) = \{s \in E^* \mid (\exists\sigma \in T^*)\ M_0[\sigma\rangle M, \ell(\sigma) = s\}$.

Given two LPNs $G_1 = (N_1, M_{0,1}, E_1, \ell_1)$ with $N_1 = (P_1, T_1, Pre_1, Post_1)$ and $G_2 = (N_2, M_{0,2}, E_2, \ell_2)$ with $N_2 = (P_2, T_2, Pre_2, Post_2)$, the *concurrent composition* of G_1 and G_2 is a LPN $G = G_1 \parallel G_2 = (N, \begin{bmatrix} M_{0,1} \\ M_{0,2} \end{bmatrix}, E_1 \cup E_2, \ell)$ with $N = (P, T, Pre, Post)$, $P = P_1 \cup P_2$, and T, Pre and $Post$ defined as follows:

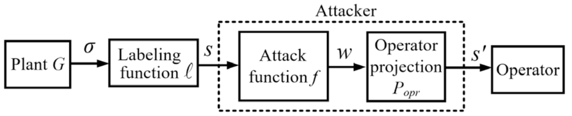

Fig. 1. Attack model.

$$\begin{cases}
t_1 \in T_1,\ \ell_1(t_1) \in (E_1 \setminus E_2) \cup \{\varepsilon\} \Rightarrow \\
\quad (t_1, \varepsilon) \in T,\ \ell[(t_1, \varepsilon)] = \ell_1(t_1), \\
\quad Pre[(\cdot, (t_1, \varepsilon))] = \begin{bmatrix} Pre_1(\cdot, t_1) \\ \mathbf{0}_{|P_2|\times 1} \end{bmatrix}, \\
\quad Post[(\cdot, (t_1, \varepsilon))] = \begin{bmatrix} Post_1(\cdot, t_1) \\ \mathbf{0}_{|P_2|\times 1} \end{bmatrix}, \\
t_2 \in T_2,\ \ell_2(t_2) \in (E_2 \setminus E_1) \cup \{\varepsilon\} \Rightarrow \\
\quad (\varepsilon, t_2) \in T,\ \ell[(\varepsilon, t_2)] = \ell_2(t_2), \\
\quad Pre[(\cdot, (\varepsilon, t_2))] = \begin{bmatrix} \mathbf{0}_{|P_1|\times 1} \\ Pre_2(\cdot, t_2) \end{bmatrix}, \\
\quad Post[(\cdot, (\varepsilon, t_2))] = \begin{bmatrix} \mathbf{0}_{|P_1|\times 1} \\ Post_2(\cdot, t_2) \end{bmatrix}, \\
t_1 \in T_1,\ t_2 \in T_2,\ \ell_1(t_1) = \ell_2(t_2) \in E_1 \cap E_2 \Rightarrow \\
\quad (t_1, t_2) \in T,\ \ell[(t_1, t_2)] = \ell_1(t_1), \\
\quad Pre[(\cdot, (t_1, t_2))] = \begin{bmatrix} Pre_1(\cdot, t_1) \\ Pre_2(\cdot, t_2) \end{bmatrix}, \\
\quad Post[(\cdot, (t_1, t_2))] = \begin{bmatrix} Post_1(\cdot, t_1) \\ Post_2(\cdot, t_2) \end{bmatrix}.
\end{cases} \tag{1}$$

2.3 Deterministic Finite State Automata

A *deterministic finite state automaton* is a four tuple $G = (X, \Sigma, \xi, x_0)$, where X is the set of states, Σ is the set of events, $\xi : X \times \Sigma \to X$ is the transition function, and x_0 is the initial state.

3. ATTACK MODEL

In this paper, we consider a plant modeled by LPN. As sketched in Fig. 1, the plant generates a *sequence of transitions* $\sigma \in T^*$, and a *sequence of labels* $s \in E^*$ obtained via labeling function ℓ. The attacker may corrupt s by inserting or erasing some sensor readings of the operator, and the operator constructs its *state estimation* based on the *sequence of corrupted labels* $s' \in E^*$. It may happen that a critical marking is reached by executing σ, but the operator evaluates the plant is in a noncritical marking according to s'. In such a case, the plant should be protected, but the operator activates no protection. Thus, damages may occur on the system (Neglect for the moment the internal structure of an attacker).

The set of *compromised labels* is denoted as $E_{com} \subseteq E$. It denotes the set of labels that the attacker can insert to the operator observation, or erase from the operator observation. The set of compromised labels can be partitioned into two subsets, namely the set of labels that can be inserted E_{ins}, and the set of labels that can be erased E_{era}. We assume that E_{ins} and E_{era} are not necessarily disjoint.

The set of *inserted labels* is denoted as $E_+ = \{e_+ \mid e \in E_{ins}\}$. The occurrence of $e_+ \in E_+$ means that the attacker inserts to the operator observation a label e.

The set of *erased labels* is denoted as $E_- = \{e_- \mid e \in E_{era}\}$. The occurrence of $e_- \in E_-$ means that the attacker erases the observation of e. Finally, the set of *attack labels* is defined by $E_a = E \cup E_+ \cup E_-$, and we assume that E, E_+, and E_- are disjoint.

Note that E_{ins} and E_{era} are the original sets of labels of the plant belonging to the set of labels E. On the contrary, E_+ and E_- are new sets of labels generated by the attacker.

Definition 1. Consider a plant G with set of compromised labels E_{com}. An attacker can be defined as an *attack function* $f : \mathcal{L}(G) \to E_a^*$:

(1) $f(\varepsilon) \in E_+^*$,
(2) $f(se) \in f(s)\{e_-, e\}E_+^*$ if $e \in E_{era}$,
(3) $f(se) \in f(s)eE_+^*$ if $e \in E \setminus E_{era}$,

where $s \in E^*$, and E_+ is the set of inserted labels. ◇

In Definition 1, condition (1) means that the attacker can insert any sequence of labels in E_+^* even though no label has been generated by the plant. Condition (2) indicates that if a transition labeled $e \in E_{era}$ fires, the attacker may either erase e or keep it unaltered, then it may insert any sequence of labels in E_+^*. Condition (3) implies that if a transition labeled $e \in E \setminus E_{era}$ fires, the attacker cannot erase e, but it can insert any sequence of labels in E_+^* after e.

The *attack language* of G is defined as $\mathcal{L}(f, G) = f(\mathcal{L}(G))$, and the *attack sequence of labels* is denoted as $w \in \mathcal{L}(f, G) \subseteq E_a^*$.

Definition 2. Consider a plant with set of compromised labels E_{com}. The *attacker projection* is defined as $P_{att} : E_a^* \to E^*$:

$$P_{att}(\varepsilon) = \varepsilon,\ P_{att}(we') = \begin{cases} P_{att}(w) \text{ if } e' = e_+ \in E_+, \\ P_{att}(w)e \text{ if } e' = e \in E \text{ or} \\ \qquad\qquad e' = e_- \in E_-. \end{cases} \quad (2)$$

The *operator projection* is defined as $P_{opr} : E_a^* \to E^*$:

$$P_{opr}(\varepsilon) = \varepsilon,\ P_{opr}(we') = \begin{cases} P_{opr}(w) \text{ if } e' = e_- \in E_-, \\ P_{opr}(w)e \text{ if } e' = e \in E \text{ or} \\ \qquad\qquad e' = e_+ \in E_+. \end{cases} \quad (3)$$

◇

By Definition 2, the attacker projection describes how the attacker interprets the sequences of labels in E_a. This affects the way it updates the marking. Since the attacker knows that $e_+ \in E_+$ are fake labels, it does not update its marking when e_+ is generated. Since the attacker knows that $e_- \in E_-$ are labels that have been erased, when e_- is generated, the attacker updates its marking the same way it does when e is generated.

The operator projection describes how the operator interprets the sequences of labels in E_a. Since the operator cannot observe labels in E_-, then it does not update its marking when $e_- \in E_-$ is generated. Since the operator cannot distinguish e from e_+, then, when e_+ is generated, the operator updates its marking the same way it does when e is generated.

The internal structure of an attacker is represented in Fig. 1 within the dashed lines. It can be seen as a black box having an observed string s as an input and generating a corrupted observation s' as an output. The intermediate word w belongs to the attack language $\mathcal{L}(f, G) \subseteq E_a^*$.

In this work, given a plant $G = (N, M_0, E, \ell)$, and a set of critical markings $\mathcal{M}$, the following assumption is made:

(A1) $\nexists \sigma,\ \sigma' \in L(N, M_0),\ \ell(\sigma) = \ell(\sigma')$: $M_0[\sigma\rangle M,\ M_0[\sigma'\rangle M',$ $M \in \mathcal{M},\ M' \notin \mathcal{M}$.

Assumption (A1) guarantees that the plant does not contain two sequences of transitions that produce the same observation, one leading to a critical marking, the other one leading to a noncritical marking. This implies that the operator can always establish if the plant is in a critical marking or not when no attack occurs. This assumption clearly simplifies our problem.

Definition 3. Consider a plant G with set of compromised labels E_{com}. Let $\mathcal{M}$ be a set of critical markings. An operator observes the plant to establish if a set of critical markings is reached. An attack function is said to be:

- *potentially effective* if $\exists w \in \mathcal{L}(f, G), \exists \sigma \in \ell^{-1}[P_{att}(w)] \cap L(N, M_0), \exists \sigma' \in \ell^{-1}[P_{opr}(w)] \cap L(N, M_0)$: $M_0[\sigma\rangle M$, $M_0[\sigma'\rangle M'$, $M \in \mathcal{M}$, $M' \notin \mathcal{M}$,

where $\mathcal{L}(f, G)$ is the attack language of G, $\sigma \in T^*$, $\sigma' \in T^*$, and $L(N, M_0)$ is the set of all sequences of transitions that can be fired at the initial marking M_0. ◇

By Definition 3, an attack function is potentially effective if a critical marking can be reached by firing a sequence of transitions producing the observation $P_{att}(w)$, and a noncritical marking can be reached by firing a sequence of transitions producing the observation $P_{opr}(w)$. Indeed, this means that, when w is generated, the plant reaches a critical marking, while the operator evaluates the plant is in a noncritical marking.

Note that, Assumption (A1) ensures that if a critical marking can be reached by firing a sequence of transitions producing the observation $P_{att}(w)$, then all the other sequences of transitions (if any) producing the observation $P_{att}(w)$ also lead to a critical marking; analogously, if a noncritical marking can be reached by firing a sequence of transitions producing the observation $P_{opr}(w)$, then all the other sequences of transitions (if any) producing the observation $P_{opr}(w)$ also lead to a noncritical marking.

Consider a plant G with set of compromised labels E_{com}. An attack function is said to be *effective* if it is potentially effective and it remains stealthy, i.e., the condition in Definition 3 is satisfied and the operator should not be able to detect the plant is under attack. The appropriate policies for an attacker to remain stealthy will be discussed in Subsection 6.2.

4. ATTACKER MONITOR

The attacker monitor G_{att} is a labeled Petri net system that generates the language on alphabet E_a that can be produced by the plant under attack. For each word w the markings reachable generating such a word are consistent with the state estimation of the attacker based on the observation $P_{att}(w) \in E^*$. The attacker monitor can be constructed using Algorithm 1.

In the following, to keep the notation more concise, we denote as $t(e)$ (resp., $t(\varepsilon)$, $t(e_+)$, $t(e_-)$) the transition t labeled $e \in E$ (resp., ε, $e_+ \in E_+$, $e_- \in E_-$). In addition, we denote as $T(E_+) = \{t(e_+) \mid e \in E_{ins}\}$ (resp., $T(E_-) = \{t(e_-) \mid e \in E_{era}\}$) the set of transitions labeled with a symbol in E_+ (resp., E_-).

Algorithm 1 Construction of the attacker monitor G_{att}

```
Input: A plant $G = (N, M_0, E, \ell)$ with $N = (P, T, Pre, Post)$, and $E_{com}$.
Output: An attacker monitor $G_{att} = (N_{att}, M_0, E_a, \ell_{att})$ with $N_{att} = (P, T_{att}, Pre_{att}, Post_{att})$.
 1: for all $t \in T$, do
 2:     $Pre_{att}(\cdot, t) = Pre(\cdot, t)$;
 3:     $Post_{att}(\cdot, t) = Post(\cdot, t)$;
 4: end for
 5: Let $T(E_-) = \emptyset$;
 6: for all $t_i \in T$: $\ell(t_i) = e \in E_{era}$ do,
 7:     $Pre_{att}(\cdot, t'_i(e_-)) = Pre(\cdot, t_i)$;
 8:     $Post_{att}(\cdot, t'_i(e_-)) = Post(\cdot, t_i)$;
 9:     $T(E_-) = \{t'_i(e_-)\} \cup T(E_-)$;
10: end for
11: Let $T(E_+) = \emptyset$;
12: for all $e \in E_{ins}$, do
13:     $Pre(\cdot, t'(e_+)) = Post(\cdot, t'(e_+)) = \mathbf{0}_{|P|\times 1}$;
14:     $T(E_+) = \{t'(e_+)\} \cup T(E_+)$;
15: end for
16: Let $T_{att} = T \cup T(E_+) \cup T(E_-)$;
17: Let $E_a = E \cup E_+ \cup E_-$.
```

We briefly explain how Algorithm 1 works. By Steps 1–4, Pre_{att} and $Post_{att}$ of G_{att} associated with $p \in P$ and $t \in T$ are the same with Pre and $Post$ of G.

At Step 5, $T(E_-)$ is initialized at $\emptyset$. By Steps 6–10, Pre_{att} (resp., $Post_{att}$) associated with $p \in P$ and t'_i labeled e_- is taken equal to Pre (resp., $Post$) associated with $p \in P$ and t_i labeled $e \in E_{era}$. This follows from the fact that the attacker knows that $e_- \in E_-$ is the label of a transition that has occurred in the plant, but has been erased by itself, so it treats e_- the same way as the real label e in the plant. At Step 9, we add $t'_i(e_-)$ to the set $T(E_-)$.

At Step 11, $T(E_+)$ is initialized at $\emptyset$. From Steps 12–15, for each $e \in E_{ins}$, we add a transition t' labeled e_+ to the set $T(E_+)$, and we impose $Pre(\cdot, t'(e_+)) = Post(\cdot, t'(e_+)) = \mathbf{0}_{|P|\times 1}$. Since the attacker knows that labels $e_+ \in E_+$ are fake labels inserted by itself, it adds no pre and post arcs connected with $t'(e_+)$, in order to avoid that the marking changes when such a transition fires. Therefore, transitions labeled e_+ correspond to transitions that have no input and output places in the attacker monitor.

Steps 16–17 define the set of transitions of the attacker monitor, namely, $T_{att} = T \cup T(E_+) \cup T(E_-)$, and its alphabet $E_a = E \cup E_+ \cup E_-$.

Example 4. Consider a plant modeled by the LPN $G = (N, M_0, E, \ell)$ in Fig. 2, where $\ell(t_1) = a$, $\ell(t_2) = b$, $\ell(t_3) = \varepsilon$, and $M_0 = p_1 + p_2$. Assume that $E_{ins} = \{b\}$ and $E_{era} = \{a\}$. The attacker monitor resulting from Algorithm 1 is sketched in Fig. 3.

Since $a \in E_{era}$, and there exist arcs directed from p_1 to $t_1(a)$ and from $t_1(a)$ to p_2 in the plant, then we add arcs directed from p_1 to $t'_1(a_-)$ and from $t'_1(a_-)$ to p_2 in the attacker monitor. Since $b \in E_{ins}$, then we add a transition labeled b_+ with no input and output place denoted as $t'(b_+)$ in Fig. 3. ◇

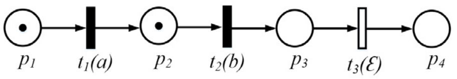

Fig. 2. Plant G.

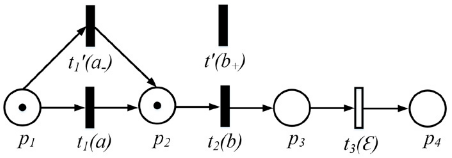

Fig. 3. Attacker monitor G_{att}.

The set of attack functions is denoted by $\mathcal{F}$, and the set of all the attack languages $\mathcal{L}(\mathcal{F}, G)$ is defined as:

$$\mathcal{L}(\mathcal{F}, G) = \bigcup_{f \in \mathcal{F}} \mathcal{L}(f, G) = \bigcup_{f \in \mathcal{F}} f(\mathcal{L}(G)). \tag{4}$$

The following proposition provides a characterization of the generated language of the attacker monitor G_{att}.

Proposition 5. Consider a plant G with set of compromised labels E_{com}. Let G_{att} be the attacker monitor constructed using Algorithm 1. It holds that:

$$\mathcal{L}(G_{att}) = \mathcal{L}(\mathcal{F}, G). \tag{5}$$

Proof. According to Algorithm 1, Steps 1–4 imply that all the uncorrupted words belong to $\mathcal{L}(G_{att})$. Steps 6–10 indicate that all attacks resulting from the erasure of labels in E_{era} are taken into account. Finally, Steps 12–15 guarantee that all attacks resulting from the insertion of labels in E_{ins} are considered. □

5. OPERATOR MONITOR

The operator monitor G_{opr} is a labeled Petri net system that generates a subset of the words $w \in E_a^*$ that can be produced by the plant under attack. In particular only words $w \in E_a^*$ whose operator projection $P_{opr}(w) \in E^*$ is a word in the language of the plant without attack can be generated. For each word w the markings reachable generating such a word are consistent with the state estimation of the operator based on the observation $P_{opr}(w) \in E^*$. The operator monitor can be constructed using Algorithm 2.

We briefly explain how Algorithm 2 works. By Steps 1–4, Pre_{opr} and $Post_{opr}$ of G_{opr} associated with $p \in P$ and $t \in T$ are the same with Pre and $Post$ of G.

At Step 5, set $T(E_+)$ is initialized at $\emptyset$. By Steps 6–10, Pre_{opr} (resp., $Post_{opr}$) associated with $p \in P$ and t''_i labeled e_+ is taken equal to Pre (resp., $Post$) associated with $p \in P$ and t_i labeled $e \in E_{ins}$. Since the operator cannot distinguish e from e_+, it treats e_+ the same way as the real label e. By Step 9, we add $t''_i(e_+)$ to the set $T(E_+)$.

By Step 11, set $T(E_-)$ is initialized at $\emptyset$. From Steps 12–15, for each $e \in E_{era}$, we add transition t'' labeled e_- to the set $T(E_-)$, and we impose $Pre(\cdot, t''(e_-)) = Post(\cdot, t''(e_-)) =$

Algorithm 2 Construction of the operator monitor G_{opr}

```
Input: A plant $G = (N, M_0, E, \ell)$ with $N = (P, T, Pre, Post)$, and $E_{com}$.
Output: An operator monitor $G_{opr} = (N_{opr}, M_0, E_a, \ell_{opr})$ with $N_{opr} = (P, T_{opr}, Pre_{opr}, Post_{opr})$.
1: for all $t \in T$, do
2:     $Pre_{opr}(\cdot, t) = Pre(\cdot, t)$;
3:     $Post_{opr}(\cdot, t) = Post(\cdot, t)$;
4: end for
5: Let $T(E_+) = \emptyset$;
6: for all $t_i \in T$: $\ell(t_i) = e \in E_{ins}$, do
7:     $Pre_{opr}(\cdot, t''_i(e_+)) = Pre(\cdot, t_i)$;
8:     $Post_{opr}(\cdot, t''_i(e_+)) = Post(\cdot, t_i)$;
9:     $T(E_+) = \{t''_i(e_+)\} \cup T(E_+)$;
10: end for
11: Let $T(E_-) = \emptyset$;
12: for all $e \in E_{era}$, do
13:     $Pre(\cdot, t''(e_-)) = Post(\cdot, t''(e_-)) = \mathbf{0}_{|P|\times 1}$;
14:     $T(E_-) = \{t''(e_-)\} \cup T(E_-)$;
15: end for
16: Let $T_{opr} = T \cup T(E_+) \cup T(E_-)$;
17: Let $E_a = E \cup E_+ \cup E_-$.
```

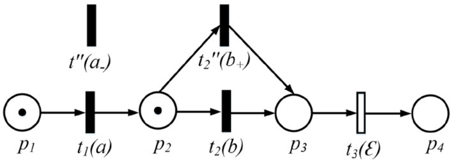

Fig. 4. Operator monitor G_{opr}.

$\mathbf{0}_{|P|\times 1}$. Since the operator cannot observe $e_- \in E_-$, then transitions labeled e_- correspond to transitions with no input and output place in the operator monitor.

Steps 16–17 define the set of transitions of the operator monitor, namely $T_{opr} = T \cup T(E_+) \cup T(E_-)$, and its alphabet $E_a = E \cup E_+ \cup E_-$.

Example 6. Consider again the plant in Fig. 2. Let $E_{ins} = \{b\}$, and $E_{era} = \{a\}$. The operator monitor is depicted in Fig. 4.

Since $b \in E_{ins}$, and there exist arcs directed from p_2 to $t_2(b)$ and from $t_2(b)$ to p_3 in the plant, then we add arcs directed from p_2 to $t''_2(b_+)$ and from $t''_2(b_+)$ to p_3 in the operator monitor. Since $a \in E_{era}$, we add a transition $t''(a_-)$ that has no input and output place. ⋄

Consider a plant G with set of compromised labels E_{com}. The set of *stealthy words* of the plant is defined as:

$$W_s = \{w \in E_a^* \mid P_{opr}(w) \in \mathcal{L}(G)\}. \tag{6}$$

The following proposition provides a characterization of the generated language of the operator monitor G_{opr}.

Proposition 7. Consider a plant G with set of compromised labels E_{com}. Let G_{opr} be the operator monitor resulting from Algorithm 2. It holds that:

$$\mathcal{L}(G_{opr}) = W_s. \tag{7}$$

Proof. In accordance to Algorithm 2, Steps 1–4 ensure that $\mathcal{L}(G_{opr})$ contains all words that can be observed when no attack happens. Steps 6–10 guarantee that the firing of transitions labeled $e_+ \in E_+$ and of transitions labeled $e \in E_{ins}$ have the same influence on the marking of G_{opr}. Steps 12–15 imply that when a transition labeled $e_- \in E_-$ fires, the marking of G_{opr} is not updated. □

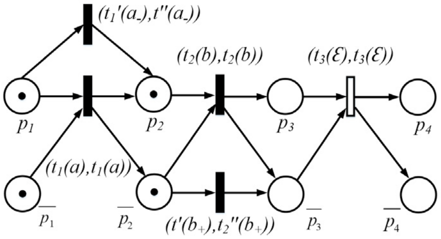

Fig. 5. Stealthy attack Petri net G_s.

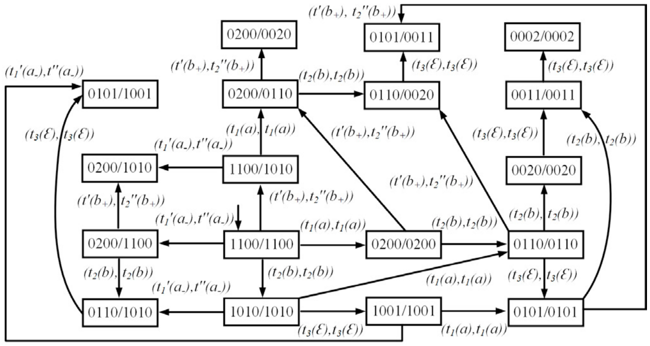

Fig. 6. Reachability graph of the stealthy attack Petri net.

6. ATTACK STRUCTURE

6.1 Stealthy Attack Petri Net

The attacker clearly wants to remain stealthy, namely, it wants to be sure that the operator does not discover its presence. To this purpose, it alters the system output guaranteeing that any sequence that the operator observes in the presence of an attack, may also be produced by the plant under no attack.

Consider a plant G with set of compromised labels E_{com}. The *stealthy attack Petri net* is a labeled Petri net system $G_s = (N_s, M_{0,s}, E_a, \ell_s)$ with $N_s = (P_s, T_s, Pre_s, Post_s)$, obtained by the concurrent composition of the attacker monitor G_{att} and the operator monitor G_{opr}.

For each word w the markings reachable generating such a word is the cartesian product of a marking consistent with the corresponding state estimation of the attacker and of the operator.

Example 8. Consider again the attacker monitor and the operator monitor sketched in Figs. 3 and 4, respectively. The stealthy attack Petri net G_s is depicted in Fig. 5, and its reachability graph is visualized in Fig. 6.

As sketched in Fig. 6, at the initial marking 1100/1100, the attacker can insert a label b, corresponding to the arc $(t'(b_+), t''_2(b_+))$ from marking 1100/1100 to marking 1100/1010. Transition $(t_1(a), t_1(a))$ may fire at marking 1100/1100. The attacker may erase it, corresponding to arc $(t'_1(a_-), t''(a_-))$ in the reachability graph in Fig. 6 from marking 1100/1100 to marking 0200/1100. The attacker does not erase it, this corresponds to the arc $(t_1(a), t_1(a))$ from marking 1100/1100 to marking 0200/0200. Transition $(t_2(b), t_2(b))$ may fire at marking 1100/1100, corresponding to the arc $(t_2(b), t_2(b))$ from 1100/1100 to 1010/1010. ◇

Theorem 9. Consider a plant G with set of compromised labels E_{com}. Let G_s be the stealthy attack Petri net. It holds that:

$$\mathcal{L}(G_s) = \mathcal{L}(\mathcal{F}, G) \cap W_s. \tag{8}$$

Proof. Since the stealthy attack Petri net G_s is obtained by doing the concurrent composition of the attacker monitor G_{att} and the operator monitor G_{opr} that have the same alphabet E_a, thus $\mathcal{L}(G_s)$ equals to the intersection of $\mathcal{L}(G_{att})$ and $\mathcal{L}(G_{opr})$. Therefore, by Propositions 5 and 7, it follows that $\mathcal{L}(G_s) = \mathcal{L}(\mathcal{F}, G) \cap W_s$. □

6.2 Attack Structure

Based on the reachability graph of the stealthy attack Petri net G_s, an automaton called *attack structure* G_a that contains all the possible attacks can be obtained using Algorithm 3.

Algorithm 3 Construction of the attack structure $G_a = (X, \Sigma, \xi, x_0)$

```
Input: A plant $G = (N, M_0, E, \ell)$ with $N = (P, T, Pre, Post)$, an attacker monitor $G_{att} = (N_{att}, M_0, E_a, \ell_{att})$ with $N_{att} = (P, T_{att}, Pre_{att}, Post_{att})$, an operator monitor $G_{opr} = (N_{opr}, M_0, E_a, \ell_{opr})$ with $N_{opr} = (P, T_{opr}, Pre_{opr}, Post_{opr})$, a stealthy attack Petri net $G_s = (N_s, M_{0,s}, E_a, \ell_s)$ with $N_s = (P_s, T_s, Pre_s, Post_s)$.
Output: An attack structure $G_a = (X, \Sigma, \xi, x_0)$.
1: Let $G_a = R(N_s, M_{0,s})$;
2: for all $x = (M_{att}, M_{opr}) \in X$, do
3:     for all $t_s = (t_{att}, t_{opr}) \in T_s$, do
4:         if $M_{opr} < Pre_{opr}(\cdot, t_{opr}) \wedge \ell_{att}(t_{att}) = \ell_{opr}(t_{opr}) \in E_+$, then
5:             $\xi(x, (t_{att}, t_{opr})) = (M_{att}, M_\emptyset)$;
6:             $X = \{(M_{att}, M_\emptyset)\} \cup X$;
7:         end if
8:         if $M_{opr} < Pre_{opr}(\cdot, t_{opr}) \wedge M_{att}[t_{att}\rangle M'_{att}$, then
9:             $\xi(x, (t_{att}, t_{opr})) = (M'_{att}, M_\emptyset)$;
10:            $X = \{(M'_{att}, M_\emptyset)\} \cup X$;
11:        end if
12:    end for
13: end for
```

We briefly explain how Algorithm 3 works. At Step 1, the attack structure is initialized at the reachability graph of G_s. By Steps 2–7, for each state $x = (M_{att}, M_{opr})$ and for each transition $t_s = (t_{att}, t_{opr})$, if t_{att} and t_{opr} are both labeled with a symbol in E_+, and the transition t_{opr} is not enabled in the operator monitor, then we impose $\xi(x, (t_{att}, t_{opr})) = (M_{att}, M_\emptyset)$ in the attack structure, and we add state $(M_{att}, M_\emptyset)$ to X. This implies that the attacker can insert labels in E_{ins} whenever possible.

From Steps 8–13, if the transition t_{opr} is not enabled by the operator monitor, and the firing of t_{att} leads to M'_{att} in the attacker monitor, then we impose $\xi(x, (t_{att}, t_{opr})) = (M'_{att}, M_\emptyset)$ in the attack structure, and we add state $(M'_{att}, M_\emptyset)$ to X. This occurs when the firing of transition t_{att} leads to a new marking in the attacker monitor and the transition t_{opr} is not enabled in the operator monitor.

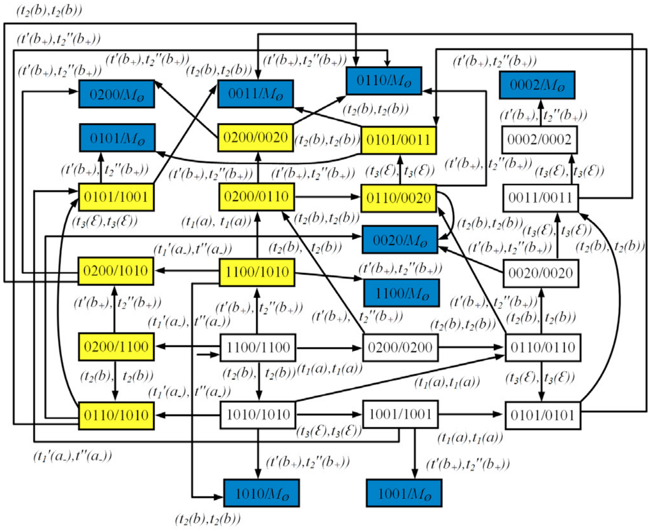

Fig. 7. Attack structure G_a. Exposing states are in blue. The states in the weakly exposing region that are not exposing states, are in yellow.

In an attack structure, let $\mathcal{M}$ be a set of critical markings. The set of *target states* $X_t = \{x = (M_{att}, M_{opr}) \mid M_{att} \in \mathcal{M}, M_{opr} \notin \mathcal{M}\}$ contains states whose first element is a critical marking and the second element is a noncritical marking. The plant reaches the critical marking while the operator evaluates the plant is in a noncritical marking when such a state is reached.

In the attack structure G_a, the set of *exposing states* $X_e = \{x = (M_{att}, M_{opr}) \mid M_{opr} = M_\emptyset\}$ contains states whose second element equals to $M_\emptyset$. The operator knows the plant is under attack when such a state is reached. Exposing states should be removed if the attacker wants to remain stealthy.

However, further additional states should be removed from the attack structure, namely those from which an exposing state will be finally reached following the evolutions of the plant. We call the set of such states *weakly exposing region* X_w.

The *supremal stealthy attack substructure* G_a^{ss} is obtained removing from the attack structure G_a all states in X_w and their input and output arcs. The set X_w can be computed using an algorithm that similar to Algorithm 3 in (Zhang et al., 2019b). The referenced paper is devoted to the analysis of cyber-attacks for automata models and thus its results can also be applied to the attack structure obtained from a stealthy attack Petri net.

In the following we will discuss this issue just by means of an example.

Example 10. Consider again the stealthy attack Petri net G_s sketched in Fig. 5. The attack structure G_a is visualized in Fig. 7. An operator observes the plant evolutions to establish if the critical marking $M = 2p_2$ is reached.

In the attack structure, there exist four target states, i.e., 0200/0020, 0200/0110, 0200/1010 and 0200/1100 (such states are colored in yellow in Fig. 7 since they belong to the weakly exposing region). When one of such states is reached, the plant is in the critical marking 0200 while the

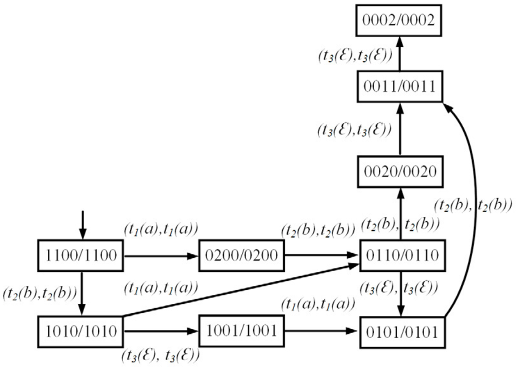

Fig. 8. Supremal stealthy attack substructure G_a^{ss}.

operator evaluates the plant is in a noncritical marking. Thus, the attack is potentially effective. For example, at the initial state 1100/1100, if transition $t_1(a)$ fires in the plant, then the attacker may erase the label a. Thus, the target state 0200/1100 is reached.

To clarify why the above states belong to the weakly exposing region, let us focus on state 0200/0020. Since there exists transition $(t_2(b), t_2(b))$ labeled $b \notin E_{era}$ that leads to the exposing state $0110/M_{\emptyset}$, and there exists no transition labeled by a symbol in E_+ that leads to a state not in X_w, then state 0200/0020 should be added to X_w. ◇

Theorem 11. Consider a labeled Petri net system G with set of compromised labels E_{com}. Let G_a be the attack structure. An attack function is potentially effective if the attack structure G_a contains a target state.

Proof. The proof follows from Definition 3 and the definition of target states. □

Theorem 12. Consider a labeled Petri net system G with set of compromised labels E_{com}. Let G_a^{ss} be the supremal stealthy attack substructure. An attack function is effective if the supremal stealthy attack substructure G_a^{ss} contains a target state.

Proof. Assume that G_a^{ss} contains a target state, this implies that the attack function is potentially effective. Furthermore, since G_a^{ss} by construction, contains no states in the weakly exposing region, then the attack function is stealthy. Thus, the attack function is effective. □

Example 13. The supremal stealthy attack substructure is sketched in Fig. 8. An operator observes the plant evolutions to establish if the critical marking $M = 2p_2$ is reached. In this special case, the supremal stealthy attack substructure equals to the reachability graph of the plant G, but in general G_a^{ss} may contain more states than the reachability graph of G. Since G_a^{ss} contains no target state, then the attack is ineffective, dually, the plant is safe. ◇

7. CONCLUSIONS AND FUTURE WORK

In this paper, the problem of stealthy sensor attacks has been investigated in the framework of LPN. We develop a supremal stealthy attack substructure that may allow the plant to reach a critical marking, while the operator evaluates that the plant is in a noncritical marking. Stealthyness of the attacker is guaranteed when the attack is defined using such a substructure. In the future, we will try to solve the problem more efficiently using GMEC and basis markings.

REFERENCES

Cabasino, M.P., Giua, A., Pocci, M., and Seatzu, C. (2011). Discrete event diagnosis using labeled Petri nets. an application to manufacturing systems. *Control Engineering Practice*, 19(9), 989–1001.

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

Giua, A., DiCesare, F., and Silva, M. (1992). Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, 974–979. IEEE.

Góes, R.M., Kwong, R., and Lafortune, S. (2019). Synthesis of sensor deception attacks for systems modeled as probabilistic automata. In *Proceedings of the 2019 American Control Conference*, 5620–5626. IEEE.

Kim, J., Lee, C., Shim, H., Eun, Y., and Seo, J.H. (2019). Detection of sensor attack and resilient state estimation for uniformly observable nonlinear systems having redundant sensors. *IEEE Transactions on Automatic Control*, 64(3), 1162–1169.

Lima, P.M., Alves, M.V.S., Carvalho, L.K., and Moreira, M.V. (2019). Security against communication network attacks of cyber-physical systems. *Journal of Control, Automation and Electrical Systems*, 30(1), 125–135.

Lin, L., Zhu, Y., and Su, R. (2019). Towards bounded synthesis of resilient supervisors against actuator attacks. *ArXiv e-prints*, [Online]. Available: https://arxiv.org/abs/1903.08358.

Ma, Z., Li, Z., and Giua, A. (2016). Petri net controllers for generalized mutual exclusion constraints with floor operators. *Automatica*, 74, 238–246.

Su, R. (2018). Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations. *Automatica*, 94, 35–44.

Tong, Y., Li, Z., Seatzu, C., and Giua, A. (2017). Verification of state-based opacity using Petri nets. *IEEE Transactions on Automatic Control*, 62(6), 2823–2837.

Wakaiki, M., Tabuada, P., and Hespanha, J.P. (2018). Supervisory control of discrete-event systems under attacks. *Dynamic Games and Applications*, [Online]. Available: https://doi.org/10.1007/s13235-018-0285-3.

Wang, Y., Bozkurt, A.K., and Pajic, M. (2019). Attack-resilient supervisory control of discrete-event systems. *ArXiv e-prints*, [Online]. Available: https://arxiv.org/abs/1904.03264.

Zhang, F., Kodituwakku, H.A.D.E., Hines, J.W., and Coble, J. (2019a). Multilayer data-driven cyber-attack detection system for industrial control systems based on network, system, and process data. *IEEE Transactions on Industrial Informatics*, 15(7), 4362–4369.

Zhang, Q., Seatzu, C., Li, Z., and Giua, A. (2019b). Cyber attacks with bounded sensor reading edits for partially-observed discrete event systems. *ArXiv e-prints*, [Online]. Available: https://arxiv.org/abs/1906.10207.

Zhu, Y., Lin, L., and Su, R. (2019). Supervisor obfuscation against actuator enablement attack. In *Proceedings of the 18th European Control Conference*, 1760–1765. IEEE.