Costs analysis of stealthy attacks with bounded output synchronized Petri nets

Rabah Ammour, Said Amari, Leonardo Brenner, Isabel Demongodin, Dimitri Lefebvre

Costs analysis of stealthy attacks with bounded output synchronized Petri nets*

Rabah Ammour[1], Said Amari[2], Leonardo Brenner[1], Isabel Demongodin[1] and, Dimitri Lefebvre[3]

***Abstract*— This paper concerns the *security analysis* of discrete event systems modeled with a particular class of synchronized Petri nets that include output functions, called Output Synchonized Petri nets. Such a formalism is suitable and tractable to represent a large variety of cyber-physical systems. In particular, we study here cyber-attacks that aim to drive the system from a given normal state to forbidden state. We assume that the attacker has a certain *credit* to insert and delete input and output events, depending on its own objectives. The proposed analysis aims to evaluate the *costs* of stealthy attacks on the controlled system depending on the objective of the controller, the structure of the system and the cost of the malicious actions.**

Keywords: Petri Nets for Automation Control, Cyber-Physical Systems, Security Analysis.

I. Introduction

Cyber-Physical Systems (CPSs) have been widely used in various real contexts: distributed automation systems, transportation networks, medical monitoring, smart power grids, advanced communication processes and vehicular social networks. This category of complex systems generally includes the plant, sensors, actuators, controllers and a communication network. CPSs arise from the interaction of physical processes, computational resources and information communication capabilities. The communication and data exchange networks between controllers and the operative part of the process are vulnerable to different types of attacks and it is common to encounter situations, where serious risks of cyber attacks occur between cyber and physical components. Examples of cyber attacks include the StuxNet strike on industrial control systems [1], and the spoofing of global positioning systems to capture unmanned aircrafts [2]. In the literature, several methodologies are devolved in the context of sensor or actuator attacks that drive a controlled Discrete Event System (DES) to unsafe or undesirable states by manipulating control and observation sequences. Studies of [3] propose a supervisor of a plant under partial observations to overcome attacks, where attacks are represented by a set-valued map that describes all possibly corrupted strings with respect to each original string. There are several results concerning the sensor attacks that drive a controlled DES to unsafe or undesirable states by manipulating observation sequences [4], [5]. Overall, these works aim to investigate one special type of cyber attacks, where an attacker can arbitrarily alter sensor readings after intercepting them from a target system in order to trick the supervisor and drives the system to an undesirable state. Works of [6] introduce a defense policy that prevents cyber attacks at sensor and actuator layer in supervisory control systems and assume that the attacker can alter the observation of events. The detectable and undetectable network attack security is established to prevent the plant from reaching forbidden states. Recently, authors of [7] exploit the techniques of the supervisory control theory to solve the actuator attacker synthesis problem. In [8], the state estimation problem for nondeterministic finite automata partially observed via a sensor measuring unit whose measurements may be vitiated by a malicious attacker, is discussed. An approach is developed to check the tamper-tolerant diagnosability of the plant by attaching attacks and costs to an enhanced version of the plant model. The context of this paper is similar to [8]. We assume that the attacker has a certain *credit* to manipulate the control symbols sent to the actuators and also the output labels returned by the sensors. We focus on the insertion and deletion of symbols and labels where each insertion or deletion has a certain *cost*. Consequently, stealthy cyber-attacks of limited cost are considered. Such attacks aim to change the system current state whatever the control sequence generated by the controller. The main contribution of the paper is to evaluate the security of the CPS depending on the credit of the attacker. For this purpose, CPSs are modeled by a class of Petri net, called an Output Synchronized Petri net (OutSynPN) [9]. This new formalism allows the system to be controlled through input events associated with controllable transitions and to be observed thanks to output labels related to some marking changes and possibly with some marking states. The logical aspects of this model, including input and output events are represented by a class of Labeled Finite state Automata with Inputs (LFAI) that lead directly to some weighted graphs useful for security analysis. The rest of the paper is organized as follows. Section II is about the motivations and backgrounds. Section III introduces the cost graph based on the LFAI. Section IV is devoted to the security analysis of CPS affected by moving attacks. Section V is a case study and Section VI concludes the paper.

*This work has been partially supported by the CPSecurity project (CNRS-INS2I grant).

[1]Rabah Ammour, Leonardo Brenner, and Isabel Demongodin are with Aix-Marseille University, University of Toulon, CNRS, LIS, Marseille, France rabah.ammour, leonardo.brenner, isabel.demongodin@lis-lab.fr

[2]Said Amari is with LURPA, ENS Paris-Saclay, France. samari@ens-paris-saclay.fr

[3]Dimitri Lefebvre is with Normandy University, GREAH, Le Havre, France. dimitri.lefebvre@univ-lehavre.fr

II. Motivations and Backgrounds

This section first discusses our motivations. A particular class of synchronized Petri nets, i.e., OutSynPNs, is then detailed. The advantage of OutSynPNs is to synchonize the transition firings on some input events and to associate some marking changes to output labels. The exhaustive logical behavior of this model is finally described as a labeled finite automaton with inputs (LFAI). The reader could find more details about Petri nets in [10], [11], [12], OutSynPNs in [9] and details about automata in [13], [14].

A. Motivations

In this paper we consider stealthy attacks that aim to drive the system from a given (normal) state to another state (that may be a forbidden state from the perspective of the system). In the spirit of covert attacks [15], the following assumptions are considered:

- the attacker knows the model of the system,
- the attacker knows (or is able to estimate) the current state of the system to perform the attack,
- the attacker can manipulate the input symbols and output labels. In particular, it can insert or delete input symbols and output labels depending on its own objectives.

Such an attack is able to change the information that circulates in both the input and output channels of the system as represented in Figure 1. Consequently, it can replace the true control sequence i by a wrong control sequence i_a. In the same time the attacker is able to erase the traces generated by its malicious actions or to insert wrong traces o_a that are similar to the expected traces o' to be observed by the user. Such attacks may become completely undetectable for the usual detection schemes that aim to compare the outputs o delivered by the system and the estimated outputs o' computed by the twin plant.

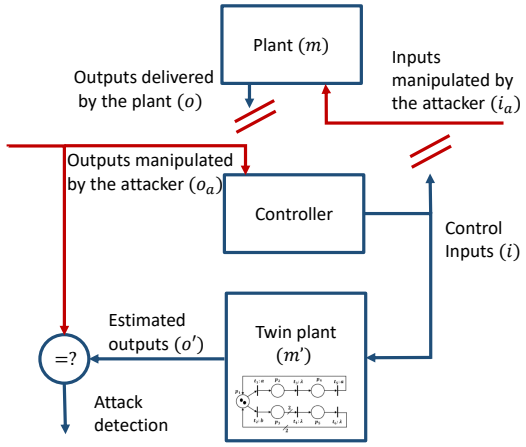

Fig. 1: Stealthy attack

B. Output Synchronized Petri nets

Let us first introduce some necessary notions. A Boolean algebra is defined on $\{0,1\}$ with conjuction, disjunction and negation operators, respectively denoted as $\wedge, \vee, \neg$. A Boolean function B is defined as $B : \{0,1\}^r \rightarrow \{0,1\}$, where r is the number of considered Boolean variables.

We denote by $\mathcal{B}_r$ the set of Boolean functions that can be defined on $\{0,1\}^r$.

Definition 1: An *output synchronized Petri net* (OutSynPN) is a structure $N_{os} = \langle N, E, f, \Sigma, \Gamma, Q, g\rangle$. A marked OutSynPN is $\langle N_{os}, M_0\rangle$ such that:

- $N = \langle P, T, Pre, Post\rangle$ is a Petri net, where P is a set of m places, T is a set of n transitions, $Pre : P \times T \rightarrow \mathbb{N}$ and $Post : P \times T \rightarrow \mathbb{N}$ are the $pre-$ and $post-$ incidence matrices that specify the weights of directed arcs from places to transitions and vice versa. $C = Post - Pre$ is the incidence matrix.
- A *marking* is a vector $M : P \rightarrow \mathbb{N}^m$ that assigns to each place a non-negative integer. We denote by $M(p)$ the marking of place p.
- E is an alphabet of external input events.
- $f : T \rightarrow E_\lambda = E \cup \{\lambda\}$ is a labeling function that associates with each transition t either an external input event $f(t) \in E$ or λ, where λ is the "always occurring" event.
- $\Sigma \subseteq \{\uparrow M(p_i), \downarrow M(p_i) \mid p_i \in P\}$ is a non empty set of conditions linked to marking changes of places, where $\downarrow$ and $\uparrow$ represent a decreasing and an increasing of a place marking, respectively.
- $\Gamma \subseteq \{M(p_i) \sim h, \mid p_i \in P, h \in \mathbb{N}, \sim \in \{=, \neq, \geq, \leq, >, <\}\}$ is a set of conditions linked to place marking values.
- Q is an alphabet of output events.
- $g : Q \rightarrow \{0,1\}$ is an output function such that $\forall q_i \in Q$, $g(q_i) = \Upsilon(B_\Gamma(q_i)) \wedge \Theta(B_\Sigma(q_i))$ where $B_\Gamma(q_i) \in \mathcal{B}_{|\Gamma|}$, $B_\Sigma(q_i) \in \mathcal{B}_{|\Sigma|}$ and:
  - $\Upsilon : \mathcal{B}_{|\Gamma|} \rightarrow \{0,1\}$ is a Boolean function depicting the conditions on the marking value of places to generate output q_i and $\Upsilon(.) = 1$ when no condition on the marking values is involved for output q_i;
  - $\Theta : \mathcal{B}_{|\Sigma|} \rightarrow \{0,1\}$ is a Boolean function depicting the conditions on the marking change events to generate output q_i and $\Theta(.) = 0$ when no event on the marking change is involved for output q_i.
- M_0 is the initial marking. ▲

In this paper, we consider that the output synchronized Petri net is *deterministic*. In other terms, transitions which are in structural conflict, do not share the same input event (i.e., $\nexists p \in P$ such that $\exists\, t, t' \in p^\bullet$, $t \neq t'$ and $f(t) = f(t')$)[1].

Let $T_e = \{t \in T \mid f(t) = e\}$ be the set of transitions associated with an input event $e \in E_\lambda$. A transition $t \in T_e$ is enabled at marking M iff $e \in E$ occurs and $M \geq Pre(\cdot, t)$. A transition $t \in T_\lambda$ is enabled at marking M iff $M \geq Pre(\cdot, t)$. At marking M, we denote by $\zeta_e(M)$ the set of enabled transitions associated with input event $e \in E$ and, by $\zeta_\lambda(M)$ the set of enabled transitions associated with the "always occurring" event λ. Thus, at marking M, transition

[1] $p^\bullet$ denotes the *postset* of place p, i.e., $p^\bullet = \{t \in T \mid Pre(p, t) > 0\}$.

$t \in T$ fires if one of following conditions is satisfied:
(i) $t \in \zeta_\lambda(M)$, i.e., transition t is enabled and $t \in T_\lambda$.
(ii) $t \in \zeta_e(M)$ and the input event $e = f(t) \in E$ occurs.

Remarks: For a transition, conditions (i) and (ii) cannot be satisfied simultaneously. For two concurrent enabled transitions, t and t', such that $t, t' \in p^\bullet$ and $t \in \zeta_\lambda(M)$, $t' \in \zeta_e(M)$ (with $e \in E$), transition t fires first. Then condition (i) has priority over condition (ii) in a certain sense.

A marking M is said to be an *unstable marking* if there exists a transition in T_λ which is enabled, i.e., $\zeta_\lambda(M) \neq \emptyset$. Otherwise the marking is a *stable marking* and is denoted by M^s in the next.

Moreover, in the Petri net theory, the server semantic is an important notion in the dynamics and must be specified. For OutSynPNs, the server semantic defines the maximal number of simultaneous firings of a given transition t for each occurrence of the input event $e = f(t)$. In the rest of this paper, we adopt a *single server semantic* [11].

The dynamics of an OutSynPN is based on an event approach with an asynchronized comportment in terms of input event and with a synchronized one in terms of firing transitions. In others terms, at a moment, only one external input event could occurs or the "always occurring" event while several enabled transitions could fire simultaneously. More precisely, given a marking M, the group of k transitions belonging to the set $\zeta_e(M) = \{t_{j_1}, \cdots, t_{j_k}\}$ with $e \in E_\lambda$, fire simultaneously when the event e occurs. This leads to the concept of *Elementary Firing Sequence (EFS)*, denoted as $\tau_e(M) = [t_{j_1}, \cdots, t_{j_k}]$, where the brackets are used to denote that the firing order of the k transitions can be arbitrary chosen since they fire simultaneously. The notation $M[\tau_e(M)\rangle M'$ means that the firing of the EFS $\tau_e(M)$ yields to marking M'. Due to the single firing semantic, one can remark that each transition $t \in T$ may appear at most one time in a given EFS. Finally, in a deterministic OutSynPN with a single server semantic, the net evolution is given, from a stable marking M_k^s, as:

1) capture the new event $e \in E$
2) fire simultaneously all enabled transitions $\zeta_e(M_k^s)$ associated with this event in a single step, i.e., $M_k^s[\tau_e(M_k^s)\rangle M'$ with $M' = M_k^s + \sum_{t \in \tau_e(M_k^s)} C(\cdot, t)$.
3) fire zero, one or more EFS $\tau_\lambda(M')$, $\tau_\lambda(M'')$, $\cdots$ corresponding to the successive enabled transitions sets $\zeta_\lambda(M'), \zeta_\lambda(M''), \cdots$ leading to the next stable marking M_{k+1}^s, i.e., $M'[\tau_\lambda(M')\rangle M''[\tau_\lambda(M'')\rangle \cdots \rangle M_{k+1}^s$.

In this paper, we deal with *bounded* OutSynPNs, i.e., for $\langle N_{os}, M_0\rangle$, it exists $k \geq 0$ such that $\forall p \in P$, $\forall M \in R(N_{os}, M_0)$, $M(p) \leq k$, where $R(N_{os}, M_0)$ is the reachability set of the place/transition net $\langle N_{os}, M_0\rangle$.

The synchronization generates constraints in the evolution of $\langle N_{os}, M_0\rangle$ compared with $\langle N, M_0\rangle$. It follows that $R(N_{os}, M_0) \subseteq R(N, M_0)$ and the set of possible firing sequences in $\langle N_{os}, M_0\rangle$ is included in the set of possible firing sequences in net $\langle N, M_0\rangle$ (i.e., the language of $\langle N_{os}, M_0\rangle$ is included in the one of $\langle N, M_0\rangle$).

An OutSynPN delivers output events. Each output event is associated with at least one marking change of a given place along with possible conditions on the marking values of one or several places. Hence, the firing of the EFS $\tau_e(M)$ may generate zero, one or more output events. Observe that the firing of $\tau_e(M)$ could generate at most one occurrence of each output event. Finally, ε is used when a given EFS does not generate any output event. Let us denote $Q(M, e) = \{q_1 \cdots q_k\} \in 2^Q$ as the set of output events generated by the firing of $\tau_e(M)$.

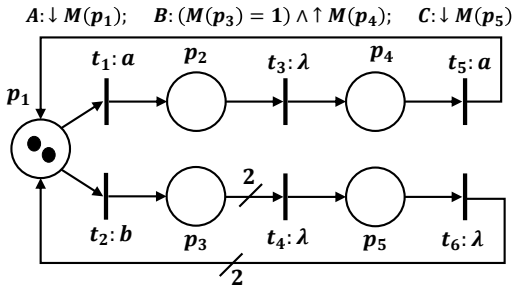

Fig. 2: An output synchronized Petri net

Example 1: Consider the OutSynPN of Figure 2 where the input/output events are $E = \{a, b\}$ and $Q = \{A, B, C\}$, respectively. The labeling function is $f(t_1) = f(t_5) = a$, $f(t_2) = b$ and $f(t_3) = f(t_4) = f(t_6) = \lambda$. Both condition sets are $\Sigma = \{\downarrow M(p_1), \uparrow M(p_4), \downarrow M(p_5)\}$ and $\Gamma = \{(M(p_3) = 1)\}$ and, with a slight abuse of notations, the output functions are: $g(A) = (\downarrow M(p_1))$, $g(B) = ((M(p_3) = 1) \wedge (\uparrow M(p_4)))$ and $g(C) = (\downarrow M(p_5))$. The initial marking is $M_0 = (2\ 0\ 0\ 0\ 0)^T$ (or $M_0 = 2p_1$).

At the initial marking, since $\zeta_\lambda(M_0) = \emptyset$, M_0 is a stable marking. Note that $\zeta_a(M_0) = \{t_1\}$ since $T_a = \{t_1, t_5\}$ but only transition t_1 respects $M_0 \geq Pre(\cdot, t_1)$. With a single server semantic, from M_0 and with the occurrence of input event a, transition t_1 fires once, i.e., $\tau_a(M_0) = [t_1]$, yielding to marking $M_1 = 1p_1, 1p_2$, i.e., $M_0[\tau_a(M_0)\rangle M_1$. The output event A is then generated, $Q(M_0, a) = \{A\}$, as there is a decreasing of the marking of p_1 (i.e., $g(A) = 1$). This new marking is unstable as $\zeta_\lambda(M_1) \neq \emptyset$ and transition t_3 fires immediately, yielding to the stable marking $1p_1, 1p_4$. No output event is generated from this firing, noted by ε. From $M_4 = 1p_1, 1p_4$, with input event a, both transitions t_1 and t_5 fire simultaneously as $\tau_a(M_4) = [t_1, t_5]$, yielding to unstable marking M_1. Thus, $M_4[\tau_a(M_4)\rangle M_1[\tau_\lambda(M_1)\rangle M_4$ with output event ε. □

In the rest of this paper, the reader will only deal with the class of bounded OutSynPNs, associated with a single server semantics and, that also satisfy the structural restriction previously presented to ensure the determinism of the model, i.e., two concurrent transitions cannot both have the same input event.

C. Labeled finite automaton with inputs

In the case of a bounded OutSynPN, the set of reachable markings is finite and the logical aspects can be represented by a particular labeled finite automaton, namely a *labeled*

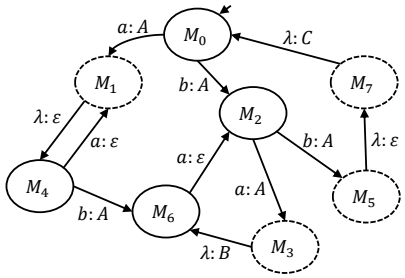

Fig. 3: LFAI of the OutSynPN in example 1

finite state automaton with inputs (LFAI) defined below.

Definition 2: A *labeled finite automaton with inputs* (LFAI) is a 6-tuple $G = (X, E_\lambda, \delta, x_0, Q, Obs)$, where

- X is a finite set of *states*,
- E is a finite set of *symbols (i.e., input events)* and $E_\lambda = E \cup \{\lambda\}$,
- $\delta : X \times E_\lambda \to X$ is a (possibly partially defined) *transition function*,
- $x_0 \in X$ is an *initial state*,
- Q is a finite set of *labels (i.e., output events)* and $Q_\varepsilon = Q \cup \{\varepsilon\}$,
- $Obs : X \times E_\lambda \to 2^Q \cup \{\varepsilon\}$ is a *labeling function*. ▲

The reachability graph of a given OutSynPN is a particular LFAI with $X = R(N_{os}, M_0)$ and $x_0 = M_0$, each state being a reachable marking M of the OutSynPN. The transition function satisfies $\delta(M, e) = M'$ if $M[\tau_e(M)\rangle M'$. The labeling function depends on the input events and also on the current state. Obs is defined such that $Obs(M, e) = Q(M, e)$ if $Q(M, e) \neq \emptyset$ and $Obs(M, e) = \varepsilon$ otherwise. There is no difficulty to extend the EFS τ, the transition function δ and the observation function Obs to any sequence of symbols $i \in E_\lambda^*$. First, observe that $\tau_{(ei)}(M) = \tau_i(M')$ with $M[\tau_e(M)\rangle M'$. Then, δ^* and Obs^* are the trivial extension of δ and Obs defined recursively by $\delta^*(M, ei) = \delta^*(M', i)$ and $Obs^*(M, ei) = Q(M, e)Obs^*(M', i)$.

Example 2: Consider the OutSynPN of Example 1. Its LFAI, given in Figure 3, is defined by a set of 8 states $X = \{M_0, \cdots, M_7\}$ that correspond to the 8 reachable markings of the OutSynPN (see Table I), including the initial state M_0. The transition and labeling functions, defined according to Definition 1, are directly derived from the OutSynPN. For example, $\delta(M_0, a) = M_1$ and $Obs(M_0, a) = A$, the same holds for the other transitions. The notation "$a : A$" means that the system switches from state M_0 to state M_1 when it receives symbol a and that this change delivers label A. The LFAI shows four unstable markings: M_1, M_3, M_5 and M_7 (see dashed circles) and four stable markings: M_0, M_2, M_4 and M_6. □

III. Cost graphs

In this section we consider that the attacker knows the model of the system, and can manipulate the symbols and

TABLE I: States of the LFAI

X	Marking	X	Marking
M_0	$(2\ 0\ 0\ 0\ 0)^T = 2p_1$	M_4	$(1\ 0\ 0\ 1\ 0)^T = 1p_1, 1p_4$
M_1	$(1\ 1\ 0\ 0\ 0)^T = 1p_1, 1p_2$	M_5	$(0\ 0\ 2\ 0\ 0)^T = 1p_3$
M_2	$(1\ 0\ 1\ 0\ 0)^T = 1p_1, 1p_3$	M_6	$(0\ 0\ 1\ 1\ 0)^T = 1p_3, 1p_4$
M_3	$(0\ 1\ 1\ 0\ 0)^T = 1p_2, 1p_3$	M_7	$(0\ 0\ 0\ 0\ 1)^T = 1p_5$

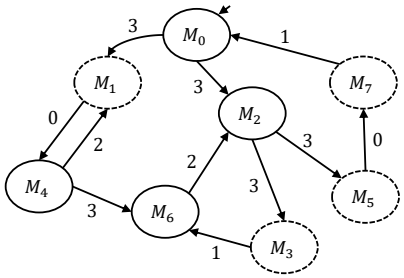

Fig. 4: Adding control graph

labels. In particular, it can insert or delete such symbols and labels depending on its own objectives. An insertion cost c_I and a deletion cost c_E are defined for each symbols and labels. Note that $c_I(\lambda) = c_E(\lambda) = 0$ and $c_I(\varepsilon) = c_E(\varepsilon) = 0$.

According to insertion and deletion costs, a weighted graph, called Adding Control Graph (ACG), can be established. It represents the cost for adding the symbol and erasing the corresponding labels for consecutive markings.

The nodes of an adding control graph are those of the LFAI while each oriented arc corresponds to a transition of the LFAI with a weight given as follows. Let $M, M' \in X$ be two states in the LFAI and $e \in E$, such that $\delta(M, e) = M'$. The weight $c_{ACG}(M, e)$ of the arc corresponding to transition $\delta(M, e) = M'$, is given by:

$$c_{ACG}(M, e) = c_I(e) + \sum_{q \in Q(M,e)} c_E(q)$$

Example 3: Consider the OutSynPN of Example 1. The cost to insert or delete each symbol and label is detailed in Table II (note that label c is not used here and will be used in the case study section).

From this table, one can compute the cost of each transition of ACG, as reported in Figure 4. For instance, the cost to drive the system from marking M_0 to marking M_1 is equal to 3 which corresponds to the sum of the cost to insert the symbol a and of the cost to erase the label A that results from the state switch, i.e., $c_{ACG}(M_0, a) = c_I(a) + c_E(A) = 2 + 1 = 3$.

□

IV. Analysis of stealthy attacks

A particular stealthy attack is considered in this section in order to illustrate the use of the graph ACG. In the next, we will consider two subsets $\mathcal{N}, \mathcal{N}' \subseteq R(N_{os}, M_0)$ composed of stable states, that are used to describe the objective of

the attacker. This objective can be formulated as to drive the system from a given state $M \in \mathcal{N}$ to another state $M' \in \mathcal{N}'$.

We consider that the attacker aims to drive the system from a given marking M to another marking M'. We assume, during a moving attack, that the controller does not send any symbol. Consequently, the attacker has no symbol to erase. The attacker inserts a wrong control sequence (i.e., a sequence of symbols) and, in the same time, he erases the observable traces (i.e., a sequence of sets of labels) that the wrong control sequence has generated.

Let us consider the sequence i_a inserted by the attacker when the current system state is M. Observe that λ-transitions may be generated spontaneously by the system and added to the attack sequence. The trajectory $\sigma(M, i'_a)$ of $k+1$ successive markings from M is defined as:

$$M_{i_0} \xrightarrow{e_1:Q(M_{i_0},e_1)} M_{i_1} \dots M_{i_{k-1}} \xrightarrow{e_k:Q(M_{i_{k-1}},e_k)} M_{i_k} \quad (1)$$

where $M_{i_0} = M$, $M_{i_k} = M'$, $i'_a = e_1 \dots e_k$, $e_h \in E_\lambda$, $h = 1, \dots, k$ is the sequence of symbols inserted by the attacker completed with the λ-transitions and $o_a = Obs^*(M, i'_a) = Q(M_{i_0}, e_1) \dots Q(M_{i_{k-1}}, e_k)$ is the corresponding sequence of sets of labels. The cost of this attack that inserts i_a at M and erases o_a can be computed as:

$$c_{MA}(M, i_a) = \sum_{e_h \in i_a} c_I(e_h) + \sum_{\substack{q \in Q(M_{i_{h-1}}, e_h) \\ Q(M_{i_{h-1}}, e_h) \in o_a}} c_E(q)$$

or equivalently as:

$$c_{MA}(M, i_a) = \sum_{(M_{i_h}, e_h) \in \sigma(M, i'_a)} c_{ACG}(M_{i_h}, e_h) \quad (2)$$

As there could exist several symbol sequences that reach marking M' from marking M, it is interesting to determine which one has the minimal cost (this sequence corresponds to the worst case from the controller perspective since the attacker could reach M' from M with the lowest cost). By using the wellknown Dijkstra algorithm, the minimal cost $c^*_{MA}(M, M')$ that the attacker needs to move the system from M to M' can be computed:

$$c^*_{MA}(M, M') = \min_i \{c_{MA}(M, i)\} \quad (3)$$

with $M[\sigma(M, i) > M'$.

More generally, it is interesting to compute the lowest cost c^*_{MA} from any state $M \in \mathcal{N}$ to any state $M' \in \mathcal{N}'$ (this last set could represent a set of deadlock states, for instance).

$$c^*_{MA} = \min_{M \in \mathcal{N}, M' \in \mathcal{N}'} \{c^*_{MA}(M, M')\} \quad (4)$$

Definition 3: Let us consider a system modeled by a marked OutSynPN $\langle N_{os}, M_0 \rangle$ and let $G = (X, E_\lambda, \delta, x_0, Q, Obs)$ be its LFAI. Let c be the credit of the attacker to move the system from $\mathcal{N}$ to $\mathcal{N}'$. The system is called *c-vulnerable to moving attack* if the minimal cost c^*_{MA} to drive the system from any state in $\mathcal{N}$ to any state in $\mathcal{N}'$ while erasing the observable traces is less than c. ▲

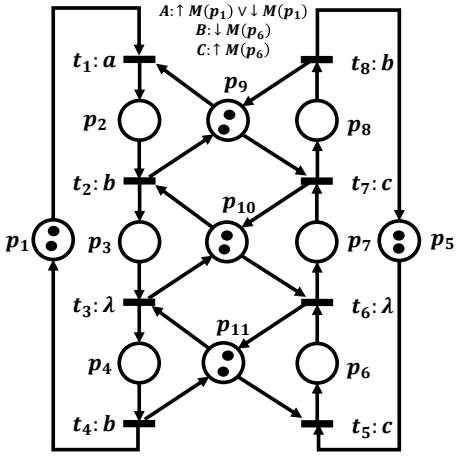

Fig. 5: OutSynPN of a jobshop

The indicator of c-vulnerability with respect to moving attacks can be interpreted as follows. In the case where this indicator is low, it means that the attacker needs only a small credit to reach a state in $\mathcal{N}'$ (e.g., a forbidden state) from at least one particular state in $\mathcal{N}$ (e.g., a normal state). In the case where this indicator is large, it means that the attacker needs numerous credits to reach any forbidden state from the set of normal states. Definition 3 will be illustrated in the case study presented in Section V.

V. Case study

The OutSynPN of Figure 5 models a jobshop with two production lines, L_1 and L_2 represented, respectively, by $p_1, t_1, p_2, t_2, p_3, t_3, p_4, t_4$ and $p_5, t_5, p_6, t_6, p_7, t_7, p_8, t_8$, where places p_1 and p_5 represent the available pallets. Three types of robots are needed to manufacture products on both lines and two robots of each type are available. These robots are represented by the resource places p_9, p_{10}, p_{11}, with two tokens in each place at initial time.

The labeling function is given by $f(t_1) = a$, $f(t_2) = f(t_4) = f(t_8) = b$, $f(t_5) = f(t_7) = c$ and $f(t_3) = f(t_6) = \lambda$. The labels are: $Q = \{A, B, C\}$ and their associated functions are given by: $g(A) = (\downarrow M(p_1)) \vee \uparrow M(p_1))$, $g(B) = (\downarrow M(p_6))$ and $g(C) = (\uparrow M(p_6))$. The initial marking, depicted on Figure 5, is $M_1 = 2p_1, 2p_5, 2p_9, 2p_{10}, 2p_{11}$, corresponding to four available pallets at initial time. The obtained LFAI has 33 states among which 19 are stable states and one is a blocking state (deadlock) corresponding to the marking $M_d = 2p_2, 2p_7, 2p_{11}$. The cost to insert or delete each symbol and label is detailed in Table II.

In the next, we consider a set $\mathcal{N}$ of states composed by stable states that use two pallets at most (i.e., $m_1 + m_5 \geq 2$, see markings given in Table III). From the controller perspective, maintaining the system state within $\mathcal{N}$ increases the security of the system by preserving unexpected transitions to M_d. Consequently, we will refer to $\mathcal{N}$ as to the set of normal states. In addition, the set $\mathcal{N}' = \{M_d\}$ is also considered. This set contains only the deadlock marking that should be

TABLE II: Cost of symbols and labels insertion and deletion

	a	b	c	A	B	C
c_I	2	2	2	3	3	3
c_E	1	1	1	1	1	1

avoided (from the perspective of the controller) and will be referred as to a forbidden state.

Let us consider an attack where the attacker aims to drive the system from any state in $\mathcal{N}$ to $\mathcal{N}'$. For instance, consider the particular case where the system is in initial marking M_0. Using the adding control graph (ACG) represented in Figure 6, where set $\mathcal{N}$ is defined by the green nodes, the minimal cost to drive the system from M_1 to M_d (red node in the ACG) is 14. It corresponds to the sequence of symbols $i_a = aacc$ that costs 8 to be inserted by the attacker. In order to hide the impact of i_a on the output channel, the attacker deletes in the meantime the sequence of labels $o_a = AACBCB$ generated by the system that costs 6. The minimal costs to drive the system from each state $M \in \mathcal{N}$ to $M_d = M_{26}$ are reported in column c^*_{MA} of Table III. The attack of minimal cost (7) corresponds to a trajectory from marking M_{11} to $M_d = M_{26}$ represented by red arrows in Figure 6. This leads to the conclusion that the system is 7-vulnerable to attacks in set $\mathcal{N}' = \{M_d\}$.

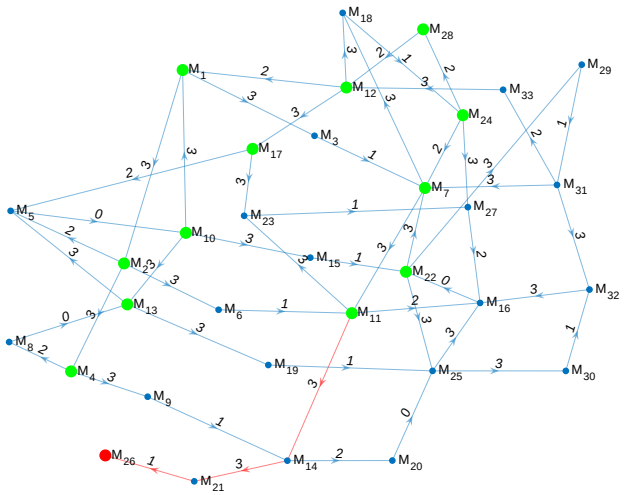

Fig. 6: Adding control graph of the case study

VI. Conclusions

This paper has proposed an approach to measure the security of cyber-physical systems modeled with Output Synchonized Petri nets that include input and output events. By computing a labeled finite automaton with inputs from the OutSynPN and a weighted graph that encode the cost of the malicious actions from the attacker, we propose indicators that evaluate the security of the controlled system. These indicators are illustrated in the particular case of attacks that aim to change the current state of the system.

TABLE III: Attack costs

State	Marking of $\mathcal{N}$	c^*_{MA} to $\mathcal{N}'$
M_1	$2p_1, 2p_5, 2p_9, 2p_{10}, 2p_{11}$	14
M_2	$1p_1, 1p_2, 2p_5, 1p_9, 2p_{10}, 2p_{11}$	11
M_4	$2p_2, 2p_5, 2p_{10}, 2p_{11}$	8
M_7	$2p_1, 1p_5, 1p_7, 2p_9, 1p_{10}, 2p_{11}$	10
M_{10}	$1p_1, 1p_4, 2p_5, 2p_9, 2p_{10}, 1p_{11}$	17
M_{11}	$1p_1, 1p_2, 1p_5, 1p_7, 1p_9, 1p_{10}, 2p_{11}$	7
M_{12}	$2p_1, 1p_5, 1p_8, 1p_9, 2p_{10}, 2p_{11}$	16
M_{13}	$1p_2, 1p_4, 2p_5, 1p_9, 2p_{10}, 1p_{11}$	20
M_{17}	$1p_1, 1p_2, 1p_5, 1p_8, 2p_{10}, 2p_{11}$	19
M_{22}	$1p_1, 1p_4, 1p_5, 1p_7, 2p_9, 1p_{10}, 1p_{11}$	13
M_{24}	$2p_1, 1p_7, 1p_8, 1p_9, 1p_{10}, 2p_{11}$	12
M_{28}	$2p_1, 2p_8, 2p_{10}, 2p_{11}$	18

In our future works we will consider more general attack scenarios. In particular, we aim to extend the proposed analysis to situations where the attacker has only a partial knowledge about the system or where it can modify the symbols and labels with some restrictions.

References

[1] J. P. Farwell and R. Rohozinski, "Stuxnet and the future of cyber war," *Survival*, vol. 53, no. 1, pp. 23–40, 2011.

[2] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via gps spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.

[3] M. Wakaiki, P. Tabuada, and J. P. Hespanha, "Supervisory control of discrete-event systems under attacks," *Dynamic Games and Applications*, vol. 9, no. 4, pp. 965–983, 2019.

[4] R. Meira-Góes, E. Kang, R. H. Kwong, and S. Lafortune, "Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems," *Automatica*, vol. 121, pp. 109–172, 2020.

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

[6] 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.

[7] L. Lin, Y. Zhu, and R. Su, "Synthesis of covert actuator attackers for free," *Discrete Event Dynamic Systems*, vol. 30, pp. 561–577, 2020.

[8] Y. Li, C. N. Hadjicostis, N. Wu, and Z. Li, "Error-and tamper-tolerant state estimation for discrete event systems under cost constraints," *arXiv preprint arXiv:2011.01371*, 2020.

[9] R. Ammour, S. Amari, L. Brenner, I. Demongodin, and D. Lefebvre, "Observer design for output synchronized Petri nets," in *European Control Conference (ECC), Netherlands, June 29 - July 2*, 2021.

[10] R. David and H. Alla, *Discrete, continuous, and hybrid Petri nets*. Springer, 2010.

[11] M. Moalla, J. Pulou, and J. Sifakis, "Synchronized Petri nets : A model for the description of non-autonomous sytems," *Mathematical Foundations of Computer Science*, pp. 374–384, 1978.

[12] M. Pocci, I. Demongodin, N. Giambiasi, and A. Giua, "Testing experiments on synchronized Petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 125–138, 2014.

[13] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[14] C. N. Hadjicostis, *Estimation and Inference in Discrete Event Systems*. Springer, 2020.

[15] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, 2015.