Observer design for labeled finite automata with inputs under stealthy actuators attacks

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

▶ **To cite this version:**

Rabah Ammour, Said Amari, Leonardo Brenner, Isabel Demongodin, Dimitri Lefebvre. Observer design for labeled finite automata with inputs under stealthy actuators attacks. 16th IFAC Workshop on Discrete Event Systems WODES, Sep 2022, Prague, Czech Republic. pp.46-51, 10.1016/j.ifacol.2022.10.322 . hal-03879792

HAL Id: hal-03879792
https://amu.hal.science/hal-03879792

Submitted on 30 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Observer design for labeled finite automata with inputs under stealthy actuators attacks ⋆

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

* *LIS, CNRS, Aix-Marseille University, France.*
** *LURPA, ENS Paris-Saclay, France.*
*** *Normandy University, GREAH, France.*

Abstract: This paper addresses the problem of cyber-attacks in discrete-event systems framework. Labeled finite state automata with inputs derived from a particular class of Petri net, called Output Synchronized Petri nets, are used to model a given cyber-physical system along with the information that circulate between controllers and plant. Stealthy cyber-attacks that may alter the control symbols, i.e., the orders sent by the controllers to the actuators, are considered. The objective is to construct an observer that uses both input and output information to provide a state estimation of the system under such stealthy actuators attacks. This observer provides a refined state estimation related to both normal and attack conditions.

Keywords: Discrete event systems, cyber-physical systems, actuator attacks, observer design.

1. INTRODUCTION

Cyber-Physical Systems (CPSs) generally include plant components, sensors, actuators, controllers and a communication network. These critical systems can be found in several industrial fields: flexible production workshops, transportation systems, distributed automation systems, medical engineering, smart power grids, and robotic systems. The growth in the use of communication networks for data exchanges, monitoring and control of physical systems increases the vulnerability of CPSs to different malicious attacks. This threat shows that the implementation of detection techniques is crucial for the reliable exploitation of such systems.

In the literature, different strategies are proposed to deal with cyber-attacks that drive a controlled Discrete Event System (DES) to unsafe or undesirable states by considering actuator attacks (Lin et al., 2020), sensor attacks (Meira-Góes et al., 2020; Su, 2017), or both (Lin and Su, 2020). These works focus on designing stealthy (i.e., undetectable) attack strategies by exploiting the existing developments in the field of supervisory control theory. By adopting an attacker viewpoint, the motivation is to provide a good understanding about the attacker' possibilities with the objective to provide insights about the system's vulnerabilities. Other works deal with attack detection and defense strategies. In (Lima et al., 2018), attacks on the sensor and/or actuator communication channels are considered and, a defense strategy is established to detect attacks or prevent damages by disabling controllable events. Zhang et al. (2019) carried out a study to analyse a supervised DES under attack. The considered attacker may corrupt the sensor reading and may also enable events that are disabled by the supervisor. A special automaton, called attack structure, is constructed as the parallel composition of the attacker observer and the supervisor under attack. The final objective is to verify the effectiveness of an attacker, i.e., its capacity to drive the plant to a critical state, while the operator estimates that the plant is in a noncritical one. Recently, the authors (Zhang et al., 2021) were interested in the state estimation problem of partially-observed DESs under sensors attack. The intruder is assumed to be able to insert or erase sensor reports. An automaton, called joint estimator, is then defined to describe the set of all possible attacks. Such structure provides, for a given sensor outputs sequence, the set of states consistent with the uncorrupted and possibly corrupted sequence. In the same context, where the attacker is able to modify a subset of sensor readings to mislead the supervisor, Meira-Góes et al. (2020), propose a bipartite transition structure, called Insertion-Deletion Attack structure which captures the game-like interaction between the supervisor and the environment. This allows one to determine if the attacker can lead the plant to a forbidden state without being detected by the supervisor.

Based on the Petri net formalism, few works have been done on CPS under stealthy attacks. Dedicated to labeled Petri nets, Zhang et al. (2020) have extended their previous work on attack structures for representing all possible sensor attacks on the system. More recently, a new Petri net formalism, called Output Synchonized Petri net (Ammour et al., 2021b), has been defined for modeling CPSs under attacks. The reachability of such a model is established by a class of automata, called labeled finite automata with inputs (LFAIs). A first study on these formalisms has been developed for vulnerability assessment of CPSs under stealthy attacks (Ammour et al., 2021a).

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

Based on these previous DES formalisms, this paper is focused on actuators attacks. We assume that the intruder can insert, replace or delete some control actions (i.e., actuator orders) depending on its own objectives but it is unable to disturb the sensor reports. In the meantime, the controller sends control inputs - referred to as symbols in the rest of this paper - to the plant and receives sensor reports from the plant in return - referred to as labels in the rest of this paper. The main contributions of the paper are: (i) to propose a composed automaton dedicated to actuator attacks where each state is formed by the assumed state by the controller (under the assumption that no attack has occurred) and the actual state (that depends on the attacker capabilities). Such an automaton provides all the possible attacker possibilities. (ii) to design an observer from the composed automaton that uses both input symbols and output labels information to provide a state estimation of the system under such stealthy actuators attacks. Such an observer provides a refined state estimation related to both normal and attack conditions.

The rest of the paper is organized as follows. In Section 2, basic definitions and terminology on LFAIs are reviewed. Section 3 is about the composed automaton for actuator attacks obtained from the composition of two LFAIs: a controller automaton and an attacker automaton. Section 4 details the design of an observer that can be used to estimate the current state of the system under normal and attack conditions. Section 5 concludes the paper.

2. LABELED FINITE AUTOMATA WITH INPUTS

In our previous work (Ammour et al., 2021b), a particular class of Petri nets, called *output synchronized Petri nets*, that allow to model the information that circulates in a given CPS has been defined. Then, in order to analyze such Petri net models, *labeled finite state automata*[1] *with inputs (LFAIs)* have been derived (Ammour et al., 2021a). In this section, we recall the definition of a LFAI and next introduce the reduced LFAI.

The states of a LFAI represent the global states of the system and the transitions are associated with input/output information.

Definition 1. A *labeled finite state 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* and $E_\lambda = E \cup \{\lambda\}$ where λ is an *internal* and "always occurring" event,
- $\delta : X \times E_\lambda \to X$ is a (partially) *transition function*,
- $x_0 \in X$ is an *initial state*,
- Q is a finite set of *labels* and $Q_\varepsilon = Q \cup \{\varepsilon\}$, where ε denotes the absence of label,
- $Obs : X \times E_\lambda \to 2^Q \cup \{\varepsilon\}$ is a *labeling function.* ▲

We consider that the LFAI is *deterministic* with respect to the symbols, i.e., $\forall x \in X, \forall e \in E_\lambda, \delta(x, e)$ is at most of cardinality 1. In such a case, we write that e is active at state x and there exist $x' \in X$ such that $x' = \delta(x, e)$. Otherwise, $\delta(x, e)$ is not defined. $\delta(x, \lambda) = x'$ means that the system will move from x to x' according to the "always occurring" event λ, i.e., without waiting for any symbol.

[1] The reader could find details about automata in (Hadjicostis, 2020).

Definition 2. Given a LFAI G, $x \in X$ is a *λ-state* if $\delta(x, \lambda)!$, i.e., $\delta(x, \lambda)$ is defined. Otherwise x is a *$\bar{\lambda}$-states.*▲

In a certain sense, λ-states are "unstable" states because the system immediately and spontaneously switches to the next state when it reaches a λ-state. Observe that when a λ-transition (i.e., a transition associated with event λ) exists from a given state x, then, it will be the unique transition outgoing from x. Thus, the set of states of a given LFAI may be split into two subsets $X = X_\lambda \cup X_R$ where X_λ is the set of λ-states that activate a λ-transition and, $X_R = X \setminus X_\lambda$ is the set of *$\bar{\lambda}$-states* that activate one or more transitions that need external inputs to fire. Note that the initial state is assumed to be a $\bar{\lambda}$-state in the rest of this paper (i.e., $x_0 \in X_R$). Multiple labels (a subset of Q) could be provided by each transition.

A control sequence of length n sent by a controller is denoted by $i = e_1 \dots e_n$ with $e_h \in E, h = 1 \dots n$. It could be completed by the λ events that are generated spontaneously by the system leading to the corresponding *executed* sequence $i' = e'_1 \dots e'_m, e'_h \in E_\lambda, h = 1 \dots m$, with $m \geq n$. As far as we restrict the discussion to deterministic LFAIs, a single i' is associated with a given i.

We introduce δ^* and Obs^* as the trivial extensions of the functions δ and Obs. These extensions are defined recursively, for an executed sequence i' by $\delta^*(x, ei') = \delta^*(\delta(x, e), i')$ and $Obs^*(x, ei') = Obs(x, e)Obs^*(\delta(x, e), i')$.

A trajectory $\sigma(x, i')$ of $m + 1$ successive states could be obtained from x as:

$$x_{j_0} \xrightarrow{(e'_1, Obs(x_{j_0}, e'_1))} x_{j_1} \dots x_{j_{m-1}} \xrightarrow{(e'_m, Obs(x_{j_{m-1}}, e'_m))} x_{j_m}$$

where $x_{j_0} = x$ and $x_{j_m} = \delta^*(x, i')$. The sequence of sets of labels generated by i' is denoted as $o = Obs^*(x, i') = Obs(x_{j_0}, e'_1) \dots Obs(x_{j_{m-1}}, e'_m)$. We use $(x_{j_{h-1}}, e'_h) \in \sigma(x, i')$ to refer to a transition from state $x_{j_{h-1}}$ driven by symbol e'_h in trajectory $\sigma(x, i')$.

The next definition introduces the notion of *reduced labeled finite automaton with inputs* (R-LFAI) that is an LFAI where all λ-states have been abstracted. Note also that a LFAI may be incomplete regarding the symbols in E. For the ease of the manipulation, the R-LFAI is defined as a complete automaton that has the same set of possible evolutions from the perspective of an outside observer.

Definition 3. Given a LFAI $G = (X, E_\lambda, \delta, x_0, Q, Obs)$ where x_0 is an initial $\bar{\lambda}$-state. Its corresponding *reduced labeled finite automaton with inputs* (R-LFAI) is a 6-tuple $G_R = (X_R, E, \delta_R, x_0, Q, Obs_R)$, where:

- X_R is a finite set of *$\bar{\lambda}$-states*,
- E is a finite set of *symbols*,
- $\delta_R : X_R \times E \to X_R$ is the *completely defined transition function* that satisfies for all $x \in X_R$:
  (i) $\delta_R(x, e) = \delta(x, e\ \lambda^k)$, $x \in X_R$, $e \in E$, if $\exists k \geq 0$, such that $\delta(x, e\ \lambda^k)!$ and $\delta(x, e\ \lambda^k) \in X_R$;
  (ii) $\delta_R(x, e) = x$, otherwise;
- $x_0 \in X_R$ is an *initial $\bar{\lambda}$-state*,
- Q is a finite set of *labels*,
- $Obs_R : X_R \times E \to 2^Q \cup \{\varepsilon\}$ is the *labeling function* that satisfies for all $x \in X_R$:
  (i) $Obs_R(x, e) = Obs(x, e\ \lambda^k)$, $x \in X_R$, $e \in E$, if $\exists k \geq 0$, such that $\delta(x, e\ \lambda^k)!$ and $\delta(x, e\ \lambda^k) \in X_R$;
  (ii) $Obs_R(x, e) = \varepsilon$ otherwise. ▲

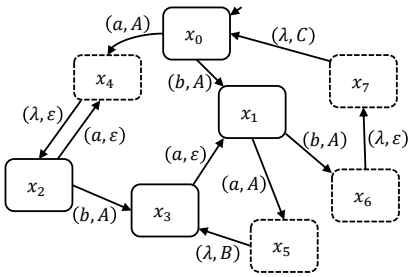

Fig. 1. A labeled finite state automaton with inputs

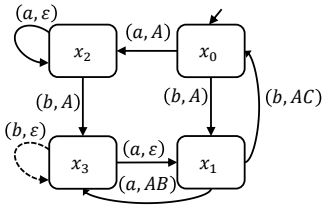

Fig. 2. R-LFAI of the LFAI in Fig. 1

Example 1. Consider the LFAI represented in Figure 1. It is defined by a set of 8 states, $X = \{x_0, \cdots, x_7\}$, that includes 4 λ-states x_4, x_5, x_6, x_7 (represented by dashed states), and the initial state x_0. The set of symbols is $E = \{a, b\}$ and the set of labels is $Q = \{A, B, C\}$. The notation (b, A) means that the system switches from state x_0 to state x_1 when it receives symbol b and, that this transition delivers label A. Thus, we have $\delta(x_0, b) = x_1$ and $Obs(x_0, b) = A$.

The R-LFAI resulting from the removing of the λ-states and from the adding of the transition $\delta_R(x_3, b) = x_3$ (in order to make the transition function completely defined with respect to the symbols generated by the controller) is reported in Figure 2. Note that the labeling function has been also completed $Obs_R(x_3, b) = \varepsilon$. □

3. COMPOSITION DESIGN

In this section, we propose to compute a logical composition that will model the impacts of any combination of actions from the controller and actions from the attacker over the communication channels between the controller and the plant. Let us recall that in this paper we consider stealthy actuator attacks which correspond to a replacement, a deletion or an insertion of control inputs.

3.1 Attack mapping

In the rest of this paper, the notation ε will be used indifferently for the absence of label or for the absence of symbol and we introduce $E_\varepsilon = E \cup \{\varepsilon\}$. The actions of the attacker are defined by a mapping f within $E_\varepsilon \times E_\varepsilon$.

Definition 4. An attack on the control symbols is defined by $f : E_\varepsilon \times E_\varepsilon \to \{0, 1\}$, such that for $e, e' \in E_\varepsilon$, $f(e, e') = 1$ if $e = e'$ or if the attacker may replace symbol e by e'; $f(e, e') = 0$, otherwise. The mapping f also specifies e'-insertion, i.e., $f(\varepsilon, e') = 1$ and e-deletion, i.e., $f(e, \varepsilon) = 1$. For each $e, e' \in E_\varepsilon$ such that $f(e, e') = 1$, e'_e denotes that the attacker replaces e by e'. ▲

Note that the mapping f could be used to represent the case where the attacker may manipulate a subset of the inputs and/or where the possible attacker actions differ depending on the considered input. In the next, we refer to the set of *attacker actions* as to E_A defined by:

$$E_A = \{e'_e \mid e, e' \in E_\varepsilon \text{ and } f(e, e') = 1\}.$$

Observe that attack actions may be separated into two groups. On one hand, *anytime actions*, i.e., symbols insertion may occur at any time whatever the actions of the controller. On the other hand, *reactive actions*, i.e., symbols replacement or deletion occurs only in reaction to some controller actions. Thus, $E_A = E_{AA} \cup E_{RA}$, where $E_{AA} = \{e'_\varepsilon \mid e' \in E \text{ and } f(\varepsilon, e') = 1\}$ and $E_{RA} = \{e'_e \mid e' \in E_\varepsilon, e \in E \text{ and } f(e, e') = 1\}$.

3.2 Controller automaton

The controller automaton describes the system evolution from the perspective of the controller. It is obtained from the R-LFAI and from the attack mapping f.

Definition 5. Given a R-LFAI $G_R = (X_R, E, \delta_R, x_0, Q, Obs_R)$ and an attack mapping f, the controller automaton $G_C(f)$ associated with G_R and f is a 6-tuple $G_C(f) = (X_R, E \cup E_A, \delta_C, x_0, Q, Obs_C)$, where:

- X_R is a finite set of $\bar{\lambda}$-*states*,
- E is the finite set of *controller symbols* and E_A is the finite set of *attacker actions*,
- $\delta_C : X_R \times (E \cup E_A) \to X_R$ is the *controller transition function* that satisfies for $x \in X_R$:
  (i) $\delta_C(x, e) = \delta_R(x, e)$ if $e \in E$;
  (ii) replacement: $\delta_C(x, e'_e) = \delta_R(x, e)$, $e'_e \in E_{RA}$, $e \in E$;
  (iii) deletion: $\delta_C(x, \varepsilon_e) = \delta_R(x, e)$, $e \in E$, $\varepsilon_e \in E_{RA}$;
  (iv) insertion: $\delta_C(x, e'_\varepsilon) = x$, $e'_\varepsilon \in E_{AA}$;
- $x_0 \in X_R$ is an *initial* $\bar{\lambda}$-*state*,
- Q is a finite set of *labels*,
- $Obs_C : X_R \times (E \cup E_A) \to 2^Q \cup \{\varepsilon\}$ is the *controller labeling function* that satisfies for $x \in X_R$:
  (i) $Obs_C(x, e) = Obs_R(x, e)$, $e \in E$;
  (ii) $Obs_C(x, e'_e) = Obs_C(x, \varepsilon_e) = Obs_R(x, e)$, $e'_e \in E_A$, $\varepsilon_e \in E_{RA}$, $e \in E$;
  (iii) $Obs_C(x, e'_\varepsilon) = \varepsilon, e'_\varepsilon \in E_{AA}$. ▲

3.3 Attacker automaton

The attacker automaton describes the system evolution from the perspective of the attacker. It is also obtained from the R-LFAI and from f.

Definition 6. Given a R-LFAI $G_R = (X_R, E, \delta_R, x_0, Q, Obs_R)$ and an attack mapping f, the attacker automaton $G_A(f)$ associated with G_R and f is a 6-tuple $G_A(f) = (X_R, E \cup E_A, \delta_A, x_0, Q_\epsilon, Obs_A)$, where:

- X_R is a finite set of $\bar{\lambda}$-*states*,
- E is the finite set of *controller symbols* and E_A is the finite set of *attacker actions*,
- $\delta_A : X_R \times (E \cup E_A) \to X_R$ is the *controller transition function* that satisfies for $x \in X_R$:
  (i) $\delta_A(x, e) = \delta_R(x, e)$ if $e \in E$;
  (ii) replacement: $\delta_A(x, e'_e) = \delta_R(x, e')$, $e'_e \in E_{RA}$, $e' \in E$;

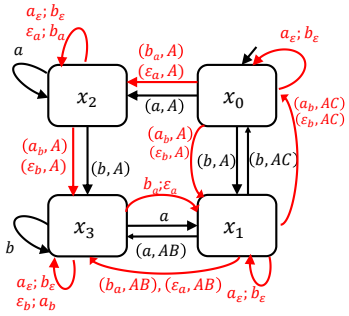

Fig. 3. Controller automaton $G_C(f)$

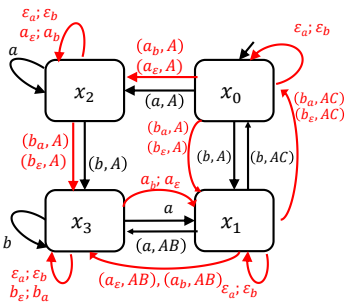

Fig. 4. Attacker automaton $G_A(f)$

(iii) deletion: $\delta_A(x, \varepsilon_e) = x$, $e \in E$, $\varepsilon_e \in E_{RA}$;
(iv) insertion: $\delta_A(x, e'_\varepsilon) = \delta_R(x, e')$, $e'_\varepsilon \in E_{AA}$;

- $x_0 \in X_R$ is an *initial λ-state*,
- Q is a finite set of *labels*,
- $Obs_A : X_R \times (E \cup E_A) \to 2^Q \cup \{\varepsilon\}$ is the *attacker labeling function* that satisfies for $x \in X_R$:
  (i) $Obs_A(x, e) = Obs_R(x, e)$, $e \in E$;
  (ii) $Obs_A(x, e'_e) = Obs_R(x, e')$, $e'_e \in E_A$
  (iii) $Obs_A(x, \varepsilon_e) = Obs_A(x, e'_\varepsilon) = \varepsilon, \varepsilon_e \in E_{RA}, e'_\varepsilon \in E_{AA}$. ▲

In the rest of the paper and for the ease of representation, an (e, ε)-transition with $e \in (E \cup E_A)$ will be represented by an arc tagged only with the symbol e.

Example 2. Consider again the LFAI in Figure 1 and its R-LFAI given in Figure 2. Consider an attack that is able to insert, erase or replace the symbols a and b. In such a case, the mapping f is given by $f(e, e') = 1$ for $e, e' \in \{a, b, \varepsilon\}$. The set of attacker actions is given by $E_A = E_{AA} \cup E_{RA}$ with $E_{AA} = \{a_\varepsilon, b_\varepsilon\}$ and $E_{RA} = \{b_a, \varepsilon_a, a_b, \varepsilon_b\}$. The controller automaton $G_C(f)$ and attacker automaton $G_A(f)$ are respectively represented in Figures 3 and 4. They both include the states and transitions of the R-LFAI but also all potential attacker's actions. In $G_C(f)$ for instance, symbols insertion could be considered at any state as long as no label is generated. These actions does not lead to any system evolution from the perspective of the controller and are represented by the self loops tagged with $a_\varepsilon; b_\varepsilon$. Symbols replacement and deletion actions are also considered and associated with the expected labels by the controller. For instance, from state x_0, (b_a, A)-transition and (ε_a, A)-transition correspond to replacement and deletion of control input a, respectively. These two transitions indicate that even if symbol a is replaced or deleted, the controller considers that state x_2 is reached as long as the expected label A is received. From the attacker's perspective who knows the real state of the system, $G_A(f)$ models all attacker's possibilities. Symbols deletion are represented by the self loops tagged with $\varepsilon_a; \varepsilon_b$ at each state. In such a case, the attacker knows that the system does not evolve and no label is generated by the system. Symbols replacement and insertion are also considered such as a_b and a_ε from x_0 that necessarily generate label A since x_0 is considered as the real state of the system. □

Note that at this stage, the feasibility of stealthy actions in $G_C(f)$ and $G_A(f)$ is not considered since they depend on both the real current state of the system (known by the attacker) and the assumed current state by the controller.

3.4 Composition design

In the next, a logical composition $G(f)$ is considered. $G(f)$ describes the interactions between the controller and the attacker from the perspective of the system. $G(f)$ is obtained by a product-like composition of $G_C(f)$ by $G_A(f)$ where the synchronisation product [2] is only defined for pairs of symbols and sets of labels that are identical in both automata. For this purpose, let us first introduce for $x, x' \in X_R$, the sets of events $\Gamma(x, x')$ and $\bar{\Gamma}(x, x')$ as:

$$\begin{aligned}\Gamma(x, x') = \{&e \in E \cup E_A : \delta_C(x, e)!, \delta_A(x', e)!,\\ &Obs_C(x, e) = Obs_A(x', e)\},\\ \bar{\Gamma}(x, x') = &(E \cup E_A) \setminus \Gamma(x, x').\end{aligned}$$

Definition 7. Let $G_R = (X_R, E, \delta_R, x_0, Q, Obs_R)$ be a R-LFAI, an attack mapping f with a set of attacker actions E_A, its associated controller automaton $G_C(f)$ and its attacker automaton $G_A(f)$. The *composed automaton* $G(f) = G_C(f) \times G_A(f)$ is the 6-tuple $G(f) = (X_{CA}, E \cup E_A, \delta_{C\times A}, s_0, Q, Obs_{CA})$, where:

- $X_{CA} = X_R \times X_R$ is the finite set of the *composition states*,
- $E \cup E_A$ is the finite set of *symbols and attacker actions*,
- $\delta_{C\times A} : X_{CA} \times (E \cup E_A) \to X_{CA}$ is a *transition function* that satisfies for $x, x' \in X_R$: $\delta_{C\times A}((x, x'), e) = (\delta_C(x, e), \delta_A(x', e))$ if $e \in \Gamma(x, x')$,
- $s_0 = (x_0, x_0) \in X_R \times X_R$ is the *initial composed automaton state*,
- Obs_{CA} is the *labeling function* that is defined for all $s = (x, x') \in X_R \times X_R$ and $e \in E \cup E_A$ as: $Obs_{CA}(s, e) = Obs_C(x, e) = Obs_A(x', e)$ if $e \in \Gamma(x, x')$. ▲

Each state s of $G(f)$ is a pair of R-LFAI states $s = (x, x')$ where x is the current state from the perspective of the controller, and x' is the system state from the perspective of the attacker that coincides with the system actual state according to the considered assumptions. Note that the assumed state by the controller x coincides with the actual state x' (i.e., $x = x'$) as far as the system is not attacked.

The set of composition states X_{CA} could be split into a set of normal states (i.e., $\{s = (x, x),\ x \in X_R\}$) and a set of states that result from undetected attacks (i.e., $\{s = (x, x'), x, x' \in X_R, x \neq x'\}$), as it is illustrated in the next example.

[2] The product of two automata $A = (X_A, E_A, \delta_A, x_{0A})$ and $B = (X_B, E_B, \delta_B, x_{0B})$ is an automaton $A \times B = (X_A \times X_B, E_A \cap E_B, \delta_{A\times B}, (x_{0A}, x_{0B}))$ where the transition function is only defined for events in the set $E_A \cap E_B$.

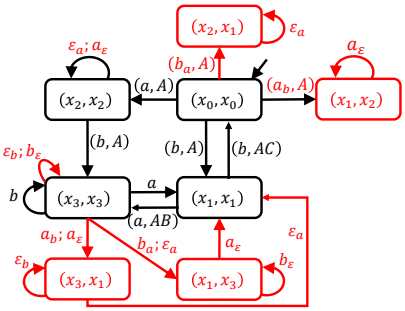

Fig. 5. Composed automaton $G(f) = G_C(f) \times G_A(f)$.

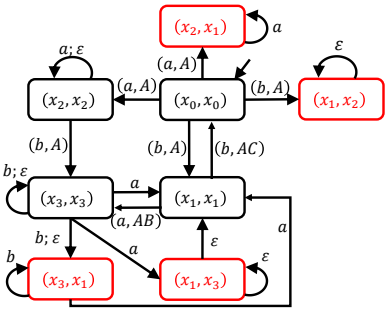

Fig. 6. Modified composition $g(G(f))$.

Example 3. The automaton $G(f)$, obtained by the composition of the controller automaton $G_C(f)$ in Figure 3 by the attacker automaton $G_A(f)$ in Figure 4, is represented in Figure 5. In this figure, the actions of the controller are associated with transitions represented with black edges, whereas the actions of the attacker are associated with transitions represented with red edges. The set of normal states is $\{(x_0,x_0),(x_1,x_1),(x_2,x_2),(x_3,x_3)\}$ whereas the one resulting from undetectable attacks is $\{(x_1,x_2),(x_2,x_1),(x_1,x_3),(x_3,x_1)\}$. For instance, from the initial state (x_0,x_0), if the controller sends symbol b and the attacker replaces it by a, then the (a_b,A)-transition drives the current state of $G(f)$ from (x_0,x_0) to (x_1,x_2). At this stage, the attacker knows that the current state has changed to x_2 whereas the controller estimates that the system state is x_1. In this situation, the attacker can continue to insert symbols a, i.e., a_ε, without being detected but this will not change the current state of the system. Any other manipulation or any symbols sent by the controller (and whatever the action of the attacker) will lead to the attack detection because the generated labels will not be consistent with the expected ones by the controller. □

It is worth noting that the composition $G(f)$ includes all combinations of attacker and controller actions. Consequently $G(f)$ can be used to evaluate the vulnerability of the system from a structural perspective, e.g., by searching connected components, paths or cycles in the graph associated to the composition. In particular, $G(f)$ is helpful to verify the existence of stealthy attacks that may lead the system to some critical situations (x,x') where the controller assumes that the state is x whereas the attacker led the system to x'.

4. OBSERVER DESIGN

In order to design an observer of the composed automaton $G(f) = G_C(f) \times G_A(f)$, let us first introduce the function g defined by: $g : E \cup E_A \to E_\varepsilon$, $g(e'_e) = e$ if $e'_e \in E_A$, and $g(e) = e$ if $e \in E$. In simple words, the function g replaces the symbols of the attacker by the symbols originally generated by the controller or by ε for anytime actions. In the rest of this paper we denote by $g(G(f))$ the structure obtained from $G(f)$ by replacing the attacker symbols by symbols in E_ε according to the function g.

Definition 8. Let $G_R = (X_R, E, \delta_R, x_0, Q, Obs_R)$ be a R-LFAI, f an attack mapping, $G_C(f)$ its associated controller automaton, $G_A(f)$ its attacker automaton and the composed automaton $G(f) = G_C(f) \times G_A(f) = (X_{CA}, E \cup E_A, \delta_{C\times A}, s_0, Q, Obs_{CA})$. The *modified composed automaton* of composed automaton $G(f)$ is the 6-tuple, $g(G(f)) = (X_{CA}, E, \delta_g, s_0, Q, Obs_g)$, where:

- $\delta_g : X_{CA} \times E_\varepsilon \to 2^{X_{CA}}$ is a transition function that satisfies: for $(x,x') \in X_{CA}$, $e \in E \cup E_A$, $\delta_{C\times A}((x,x'),e) \in \delta_g((x,x'),g(e))$.
- Obs_g is the labeling function defined by: $Obs_g((x,x'),g(e)) = Obs_{CA}((x,x'),e)$, for $(x,x') \in X_{CA}$, $e \in E \cup E_A$. ▲

An approach similar to the one already detailed in (Ammour et al., 2021b) can be used to design an observer of the modified composition $g(G(f))$. Such an observer is obtained with a slight modification of the standard approach, that transforms a non deterministic automaton into a deterministic one (Hadjicostis, 2020). Compared to the usual observers that use only the observation labels, the proposed observer takes advantage of the input / output observations of $g(G(f))$, by using both the input events generated by the controller (whatever the attacker actions) and the output labels generated by the system (that depend on the attack actions) as observations.

The initial state s_{IO0} is first defined by the set of states reachable from s_0 by executing zero or more silent transitions in $g(G(f))$ (i.e., $(\varepsilon,\varepsilon)$-transitions that will be simply refered to as ε-transitions in the next). Then, the states of the observer are successively defined by computing, for each $s \in X_{CA}$, the two following subsets of states:

- $X(s,\varepsilon) \subseteq X_{CA}$, the subset of states that are reachable from s by executing zero or more ε-transitions in $g(G(f))$;
- $X'(s,e,q) \subseteq X_{CA}$, $e \in E_\varepsilon$, $q \in Q_\varepsilon$, $(e,q) \neq (\varepsilon,\varepsilon)$, the subset of states that are reachable from s by executing exactly one (e,q)-transition in $g(G(f))$.

If for a given state s_{IO} already computed, the set $\cup_{s\in s_{IO}} \left(\cup_{s'\in X'(s,e,q)} X(s',\varepsilon)\right)$ is not empty and differs from all observer states already computed, then a new observer state s'_{IO} is created as below:

$$s'_{IO} = \cup_{s\in s_{IO}} \left(\cup_{s'\in X'(s,e,q)} X(s',\varepsilon)\right)$$

The logical observer is formally defined as follows.

Definition 9. Let $G_R = (X_R, E, \delta_R, x_0, Q, Obs_R)$ be a R-LFAI, f an attack mapping, $G_C(f)$ its associated controller automaton, $G_A(f)$ its attacker automaton and $G(f)$ the composed automaton. The *input / output observer* of the modified composed automaton

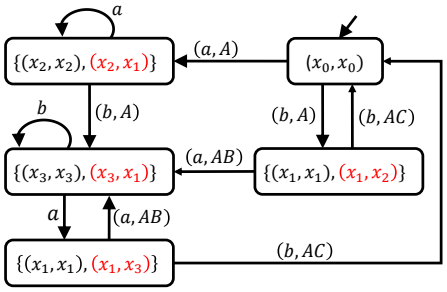

Fig. 7. Input/output observer of $g(G(f))$.

$g(G(f)) = (X_{CA}, E, \delta_g, s_0, Q, Obs_g)$, is defined by the 4-tuple $OBS_{IO} = (S_{IO}, Q_{IO}, \delta_{IO}, s_{IO0})$ with:

- $S_{IO} \subseteq 2^{X_{CA}}$, the set of observer states;
- $Q_{IO} \subseteq (E \times (2^Q \cup \{\varepsilon\}))$, the set of extended controller input symbol and associated observed label;
- δ_{IO}, the transition function defined, for all $s_{IO} \in S_{IO}$ and $(e, q) \in Q_{IO}$ by:
  $\delta_{IO}(s_{IO}, (e, q)) = \cup_{s \in s_{IO}} \left(\cup_{s' \in X'(s,e,q)} X(s', \varepsilon)\right)$ if $\cup_{s \in s_{IO}} \left(\cup_{s' \in X'(s,e,q)} X(s', \varepsilon)\right) \neq \emptyset$;
- $s_{IO0} = X(s_0, \varepsilon)$, the observer initial state. ▲

Example 4. The automaton $g(G(f))$ of Figure 6 is obtained from the composed automaton depicted in Figure 5. It includes only symbols (originally) sent by the controller even if they were manipulated by the attacker. The obtained automaton is non deterministic. For instance, from the initial state (x_0, x_0) the (a, A)-transition could lead the system to two different states (x_2, x_2) or (x_2, x_1) depending on whether the attacker has manipulated symbol a or not. ε-transitions also exist in this automaton (for instance from state (x_3, x_3) to state (x_3, x_1)) and correspond to possible system state evolution caused by the attacker without any action from the controller. Figure 7 represents the input/output observer obtained from the modified composed automaton $g(G(f))$. This observer in a deterministic automaton that indicates the states that may be reached by the system after each controller action. For instant, when the controller sends symbol b from the initial state and receives label A, then the system state could be a normal one (x_1, x_1) or a state that results from an attack (x_1, x_2). □

The observer OBS_{IO} provides valuable information to the defender. In particular, assuming that the defender knows the intruder capacities, this observer provides the system states (the second part of the pair that forms each state of $g(G(f))$) that are consistent with a given sequence of symbols and labels observed by the defender thus far. This information can serve to elaborate some defense strategy against the attacker.

In addition, by considering not only the labels but also the symbols, this observer refines the state estimation compared to a standard observer based only on the observation of the labels (for example, in the case of a system that provides few or even not any labels, and an attacker that can manipulate only a selection of symbols).

5. CONCLUSIONS

This paper has proposed a method to design a composed automaton from the perspective of the input / output channel where symbols and labels circulate between the system and the controller. Such a composition, of a controller automaton by an attacker automaton, includes all combinations of attacker and defender actions. Then, an input / output observer of the controlled system, possibly under attack, has been proposed, that makes full use of the observed symbols and labels. Such an observer provides a refined state estimation when the system is under attack.

Future works based on these approaches include several perspectives. One immediate future work is to handle more damaging attack that manipulate not only the actuators but change also the sensor reports. Another research direction is to evaluate the vulnerability of a system with respect to a given attack by proposing some formal analysis of the composed automaton and its input/output observer. Another interesting focus will be to study the question of attack and defense strategies in the perspective of optimization. Such a problem can be addressed by associating each attacker and / or controller action to a given cost related to the effort, energy or time required to perform that action.

REFERENCES

Ammour, R., Amari, S., Brenner, L., Demongodin, I., and Lefebvre, D. (2021a). Costs analysis of stealthy attacks with bounded output synchronized Petri nets. In *Int. Conf. on Autom. Science and Eng. (CASE)*. IEEE.

Ammour, R., Amari, S., Brenner, L., Demongodin, I., and Lefebvre, D. (2021b). Observer design for output synchronized Petri nets. In *European Control Conf. (ECC)*. IEEE.

Hadjicostis, C.N. (2020). *Estimation and Inference in Discrete Event Systems*. Springer.

Lima, P.M., Carvalho, L.K., and Moreira, M.V. (2018). Detectable and undetectable network attack security of cyber-physical systems. *IFAC-PapersOnLine*, 51(7), 179–185.

Lin, L. and Su, R. (2020). Synthesis of covert actuator and sensor attackers as supervisor synthesis. In *15th Int. Work. on Discrete Event Syst., WODES*. IEEE.

Lin, L., Zhu, Y., and Su, R. (2020). Synthesis of covert actuator attackers for free. *Discrete Event Dynamic Systems*, 30, 561–577.

Meira-Góes, R., Kang, E., Kwong, R.H., and Lafortune, S. (2020). Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems. *Automatica*, 121, 109172.

Su, R. (2017). A cyber attack model with bounded sensor reading alterations. In *2017 American Control Conference (ACC)*, 3200–3205.

Zhang, Q., Li, Z., and Giua, A. (2020). Stealthy sensor attacks for plants modeled by labeled Petri nets. In *15th Int. Work. on Discrete Event Syst., WODES*. IEEE.

Zhang, Q., Li, Z., Seatzu, C., and Giua, A. (2019). A framework for the analysis of supervised discrete event systems under attack. In *15th European Workshop on Advanced Control and Diagnosis (ACD)*.

Zhang, Q., Seatzu, C., Li, Z., and Giua, A. (2021). Joint state estimation under attack of discrete event systems. *IEEE Access*, 9, 168068–168079.