Discrete Event Dynamic Systems (2022) 32:143–158
https://doi.org/10.1007/s10626-021-00351-3

Discrete-event systems subject to unknown sensor attacks

Michel R. C. Alves[1] · **Patrícia N. Pena**[2] · **Karen Rudie**[3]

Received: 25 June 2021 / Accepted: 1 September 2021 / Published online: 10 December 2021

Abstract

This work is set in the context of supervisory control of discrete-event systems under partial observation. Attackers that are able to insert or erase occurrences of particular output symbols can tamper with the supervisor's observation and by doing so, can lead the controlled system to undesirable states. We consider a scenario with multiple attackers, each one being an element of a set, called the *attack set*. We also assume that only one of the attackers within an attack set is acting, although we don't know which one. According to previous results in the literature, a supervisor that enforces a given legal language, regardless of which attacker is acting, can be designed if the legal language is controllable and satisfies a property called P-observability for an attack set. The latter is an extended notion of observability and is related with the supervisor's ability to always distinguish between outputs that require different control actions, even if the outputs were attacked. We present a new approach for checking if a given language is P-observable for an attack set, by first introducing a visual representation as well as some definitions that capture the attack's effect. Additionally, we present two algorithms that together allow us to verify if a given language is P-observable for an attack set, when it is represented as an automaton.

Keywords Discrete-event system · Supervisory control · Attacks on output symbols · P-observability

✉ Michel R. C. Alves
michelrodrigo@ufmg.br

Patrícia N. Pena
ppena@ufmg.br

Karen Rudie
karen.rudie@queensu.ca

[1] Graduate Program in Electrical Engineering, Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil

[2] Department of Electronics Engineering, Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil

[3] Department of Electrical and Computer Engineering, and Ingenuity Labs Research Institute, Queen's University, Kingston, Ontario K7L 3N6, Canada

1 Introduction

In recent years, cyber-physical systems (CPSs) are being widely used in industry and are one of the key features that support the development of the industry 4.0. Cyber-physical systems are defined as embedded systems that integrate the physical and digital worlds through communication networks, providing real-time data-accessing and data-processing services (Lu 2017). The extensive use of communication networks by CPSs increases vulnerabilities for malicious attacks, thus making these networks unreliable. This concern did not exist in classical control systems (Li et al. 2020), which justifies the research effort on the subject. In order to make networks trusted, defense strategies have to be considered, which can be roughly classified as detection of attacks and prevention of the attack's effects.

In discrete-event systems (DESs), several approaches that address the problem of attacks have been proposed. In the context of supervisory control, the authors of Rashidinejad et al. (2019) provide an overview about the subject and first classify an attack as passive or active: in a passive attack, the attacker's goal is to learn a secret about the system (Lima et al. 2019), while in a active attack, the goal is to cause damage to the system (Zhang et al. 2021; Wakaiki et al. 2019; Su 2018). There are works that present methods for designing robust supervisors (Meira-Góes et al. 2021; Wang and Pajic 2019; Wakaiki et al. 2019; Lin et al. 2019 and Su 2018, among others) and the design of a detection module (as in Li et al. (2020), Gao et al. (2019), Lima et al. (2019), and Carvalho et al. (2016)). Furthermore, some authors focus on studying the design methods for the attackers, using as argument the claim that a good understanding about the adversaries can provide better insight on how to defend against them (Lin and Su 2020; Zhang et al. 2018, 2021; Mohajerani et al. 2020; Fritz and Zhang 2018; Meira-Góes et al. 2020).

There are different types of attacks considered in the literature. In Carvalho et al. (2018) the authors consider attacks in which the attacker is able to erase or insert events in both communication channels. An attack detection scheme is proposed, which only works with certainty if the system has a property called GF-safe controllability. In this work we study attacks that only happen on the output symbols and the attacker's goal is to lead the controlled system into an undesirable state. Each attacker is able to insert and/or erase symbols from a particular set. We also assume that we don't know in advance which symbols are being tampered with in the attack. Thus, in order to analyze the attack's effect, different scenarios have to be investigated. From a different perspective, some works take advantage of the ability to insert or erase events in the communication channel in a way that, under some conditions, opacity is enforced, as in Ji et al. (2019).

Differently from Carvalho et al. (2018), the authors of Wakaiki et al. (2019) proposed a design method for robust supervisors that, regardless of which symbols are under attack, always enforce the desired behavior. Two conditions are imposed over the language that represents the desired behavior: i) controllability and ii) P-observability for an attack set. The latter is a modified version of the classical notion of observability and is related to the supervisor's ability to always distinguish between observations that require different control actions. A test for checking this property was also presented, consisting of a series of pairwise classical observability tests.

In this work, we provide a new method for testing the property P-observability for an attack set. This new method actually checks for the property itself and can be done in a single run, as opposed to the multiple observability tests. Our contributions lie in the visualization technique and in the recasting of P-observability to a form that makes it easier to understand

why an attack is or is not possible and in the provision of algorithms that can be implemented to check P-observability.

The paper is organized as follows. In Section 2, we provide some basic preliminaries which are needed for understanding this work, including our attack model. In Section 3 we explain the formalization used to present the main result. The algorithms are presented in Section 4 and finally, we present the conclusions in Section 5.

2 Background on discrete event systems

In this work we adopt the usual notation and operations applied to languages and automata. For a more detailed introduction on the subject, the reader is referred to Cassandras and Lafortune (2007). In Section 2.1 we present some notation and in Section 2.2 we present the main concepts of the supervisory control theory. Finally, Section 2.3 focuses on the notation and attack model adopted in this work.

2.1 Preliminaries

An automaton G is defined as a tuple $G := (Q, \Sigma, \delta, q_0)$, where Q is a finite set of states, Σ is the nonempty finite set of events, $\delta : Q \times \Sigma \to Q$ is a partially defined transition function and q_0 is the initial state. The notation $\delta(q, \sigma)!$ represents that $\delta(q, \sigma)$ is defined for some $q \in Q$ and $\sigma \in \Sigma$. The transition function δ can be extended to a function $Q \times \Sigma^* \to Q$ according to $\delta(q, \varepsilon) := q$ and $\delta(q, s\sigma) := \delta(\delta(q, s), \sigma)$, with $q \in Q$, $s \in \Sigma^*$, $\sigma \in \Sigma$ and ε being the empty string. With abuse of notation, we sometimes treat δ as a set and $(q, \sigma, q') \in \delta$ if and only if $\delta(q, \sigma) = q'$. The automaton is said to be deterministic if $(q, \sigma, q'), (q, \sigma, q'') \in \delta$ always implies that $q' = q''$. The map $\Gamma : Q \to 2^{\Sigma}$ defined as $\Gamma(q) := \{\sigma \in \Sigma | (q, \sigma, q') \in \delta, \text{ for any } q' \in Q\}$ is the set of *feasible* events at a given state $q \in Q$. The notation 2^A, for a given set A, is the set of all subsets of A. The language generated by G, denoted by $\mathcal{L}(G)$, is defined as $\mathcal{L}(G) := \{s \in \Sigma^* | \delta(q_0, s)!\}$.

Some of the events in a DES may not be observable. Thus, the event set Σ can be partitioned as $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o is the set of *observable* events, while Σ_{uo} is the set of *unobservable* events. An observation map $P : \Sigma^* \to \Sigma_o^*$, also called *natural projection*, maps strings in Σ^* into observations in Σ_o^*. It is defined as $P(\varepsilon) := \varepsilon$, $P(\sigma) := \sigma$ if $\sigma \in \Sigma_o$, $P(\sigma) := \varepsilon$ if $\sigma \in \Sigma_{uo}$ and $P(s\sigma) := P(s)P(\sigma)$ for $s \in \Sigma^*$, $\sigma \in \Sigma$. The inverse observation map P^{-1}, called *inverse projection*, is defined as $P^{-1}(t) := \{s \in \Sigma^* | P(s) = t\}$.

2.2 Supervisory control

The set of events Σ can be also partitioned as $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where Σ_c is the set of *controllable* events and Σ_{uc} is the set of *uncontrollable* events. When G models the uncontrolled behavior of a DES, some of its states can be undesirable states, representing situations we want to avoid, as blocking or insecure operation. The legal behavior is modeled as a *desired language* $K \subseteq \mathcal{L}(G)$, that is, K is a subset of $\mathcal{L}(G)$, containing only the legal strings. We can enforce the desired language K over G by using a structure called supervisor, denoted by S, that acts over the set of controllable events. Since the supervisor cannot prevent uncontrollable events from happening, we say that a desired language K is *controllable with respect to* $\mathcal{L}(G)$ if

$$\overline{K}\,\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}.$$

If a language K is controllable, then there exists a supervisor that implements K. Formally, a supervisor is a function $S : \mathcal{L}(G) \to 2^{\Sigma}$ and for each $s \in \mathcal{L}(G)$ generated so far by G, $S(s) \cap \Gamma(\delta(q_0, s))$ is the set of *enabled* events. $S(s)$ is called *control action* at s and S is the *control policy*.

Under partial observation, the supervisor cannot distinguish between strings s_1 and s_2 if $P(s_1) = P(s_2)$. In this case, for such $s_1, s_2 \in \mathcal{L}(G)$, the supervisor must issue the same control action. A partial-observation supervisor is a function $S_P : P(\mathcal{L}(G)) \to 2^{\Sigma}$ and S_P is called a *P-supervisor*. Besides being controllable, in order to be possible to obtain a P-supervisor, the prefix-closed language K has to be *P-observable*, which is captured in the requirement that

$$\ker P \subseteq \text{act}_{K \subset L}, \tag{1}$$

where ker P denotes the relation on Σ^* defined by

$$\ker P := \{(w, w') \in \Sigma^* \times \Sigma^* | P(w) = P(w')\} \tag{2}$$

and $\text{act}_{K \subset L}$ is the binary relation on Σ^* defined by

$$\begin{aligned} \text{act}_{K \subset L} := \{&(w, w') \in \Sigma^* \times \Sigma^* | \\ &w, w' \in K \implies \nexists \sigma \in \Sigma \text{ s.t.}[w\sigma \in K, w'\sigma \in L \setminus K] \text{ or} \\ &[w\sigma \in L \setminus K, w'\sigma \in K]\}. \end{aligned} \tag{3}$$

The relation $\text{act}_{K \subset L}$ has all pairs of strings $w, w' \in K$ such that the new strings $w\sigma$ and $w'\sigma$ are either both in K or both in $L \setminus K$, for all $\sigma \in \Sigma$. Thus, even if the P-supervisor cannot distinguish between two observed strings, the control action required by them is always consistent. In the remainder of this work we call the P-supervisor simply as supervisor. Note that if P is a natural projection, then P-observability is reduced to the classical concept of observability.

2.3 Attack model

In this work we adopt the same setup as in (Wakaiki et al. 2019). Attackers whose goal is to prevent the supervisor from achieving the desired language $K \subseteq \mathcal{L}(G)$ have full observation of the communication channel between plant and supervisor and can corrupt the string of output symbols $P(w)$, $w \in \Sigma^*$ in multiple ways, by erasing and/or inserting specific output symbols and changing it to string $y \in \Sigma_o^*$. In this work, output symbols are the symbols sent from the plant to the supervisor, as an outcome of sensor readings. Also, we assume the supervisor sends a new control action to the plant whenever it receives new information. Upon reception of string y, the supervisor will then issue the control action $S(y)$. Depending on how the attacker chooses to modify $P(w)$, it can induce the supervisor to issue a control action that will make the plant reach an undesirable state.

Given a set of symbols $\alpha \subseteq \Sigma_v \subseteq \Sigma_o$ in the observation alphabet, where Σ_v is the set of *vulnerable* events, the map $R_{\neg\alpha} : \Sigma \to (\Sigma_o \cup \varepsilon)$ is called *α-removal observation map* and is defined as

$$R_{\neg\alpha}(t) := \begin{cases} \varepsilon & t \in \alpha \\ t & t \notin \alpha \end{cases} \tag{4}$$

and can be extended to a map defined for strings of output symbols in the same way as a natural projection P. An *observation attack* is a set-valued map $A_\alpha : \Sigma_o^* \to 2^{\Sigma_o^*}$ that assigns to each string $u \in \Sigma_o^*$ the set of all strings $v \in \Sigma_o^*$ that can be obtained from u by an arbitrary number of insertions or deletions of symbols in α. It is given by

$$A_\alpha(u) := \{v \in \Sigma_o^* | R_{\neg\alpha}(u) = R_{\neg\alpha}(v)\}. \tag{5}$$

A particular case happens when $\alpha = \emptyset$. In such case, $A_\emptyset$ corresponds to the absence of attack.

Thus, instead of receiving the string $P(w)$, the supervisor receives one string among the set $A_\alpha(P(w))$. The map A_α is called an *observation attack* or simply *attacker* and the map $A_\alpha P : \Sigma^* \to 2^{\Sigma_o^*}$ obtained by the composition $A_\alpha P := A_\alpha \circ P$ is the corresponding *attacked observation map*. In other words, the map $A_\alpha P(w)$ results in all strings that can be formed with the manipulation of events in $\alpha \subseteq \Sigma_o$, by inserting events into or erasing events from the observation $P(w)$. The goal of the attacker is, by changing the observation string, to make the supervisor accept a string that is not in K or to reject a string that is in K. We can also interpret the attack as a factor that increases the supervisor's uncertainty about which state the plant is really in.

We assume that an attacker knows what the plant can do and what the language K is. Therefore, if $w \in K$ is the string executed in the plant so far, then we assume that an attacker is smart enough to not insert an event $\sigma \in \alpha$ in the observation if $w\sigma$ is not a possible string in K (since otherwise, the attack would be trivially detectable).

A more interesting scenario consists of multiple attackers, instead of only one. By multiple attackers, we mean that there is an *attack set*, denoted by $\mathcal{A}$, such that $\mathcal{A} = \{A_\emptyset, A_{\alpha_1}, A_{\alpha_2}, \ldots, A_{\alpha_M}\}$, where each $\alpha_i \subseteq \Sigma_v$. We assume that only one of the attackers in an attack set $\mathcal{A}$ is acting, but we do not know which one. The authors of Wakaiki et al. (2019) present a result that gives the conditions for the existence of a supervisor that, regardless of which attacker $A_\alpha \in \mathcal{A}$ is acting, can enforce a desired language K. These conditions are *i*) controllability and *ii*) P-observability for an attack set. The latter is an extended notion of P-observability, which is given by

$$R_{A_{\alpha_i}, A_{\alpha_j}} \subset \text{act}_{K \subset L} \tag{6}$$

where the relation $R_{A_{\alpha_i}, A_{\alpha_j}}$, with $A_{\alpha_i}, A_{\alpha_j} \in \mathcal{A}$, contains all pairs of strings that may result in attacked observation maps $A_{\alpha_i} P$ and $A_{\alpha_j} P$ with a common string of output symbols, i.e.,

$$R_{A_{\alpha_i}, A_{\alpha_j}} := \{(w, w') \in \Sigma^* \times \Sigma^* | A_{\alpha_i} P(w) \cap A_{\alpha_j} P(w') \neq \emptyset\} \tag{7}$$

and $\text{act}_{K \subset L}$ is as in Eq. 3. If the language K does not have this property, it is possible to find two attackers $A_{\alpha_i}, A_{\alpha_j} \in \mathcal{A}$ that could generate the same observation $y \in \Sigma_o^*$ for two distinct strings $w, w' \in K$, and w would transition to an element in K and w' to an element outside K, or vice-versa. This means that, upon observing y, the supervisor cannot decide which control action to take. In the remainder of this work, unless said otherwise, we shall use "P-observability" as shorthand for "P-observability for an attack set".

The following result from Wakaiki et al. (2019) shows how the test for P-observability for an attack set $\mathcal{A}$ can be done by reducing it to a classical observability test.

Theorem 1 Wakaiki et al. (2019) *For every nonempty prefix-closed set $K \subseteq L$ and attack set $\mathcal{A} = \{A_{\alpha_1}, A_{\alpha_2}, \ldots, A_{\alpha_M}\}$ consisting of $M \geq 1$ observation attacks, K is P-observable for the set of attacks $\mathcal{A}$ if and only if K is $(R_{\neg\alpha} \circ P)$-observable (in the classical sense, i.e., without attacks) for every set $\alpha := \alpha_i \cup \alpha_j$, $\forall i, j \in \{1, 2, \ldots, M\}$.*

Theorem 1 states that P-observability for an attack set $\mathcal{A}$ can be tested by picking every possible pair of attackers $A_{\alpha_i}, A_{\alpha_j}$ and checking if regular observability is obtained if the symbols affected by the two attackers are removed from the observation. The authors claim that, using the test for classical observability presented in Cassandras and Lafortune (2007, Section 3.7.3), P-observability for an attack set can be tested with time complexity $O(M^5)$,

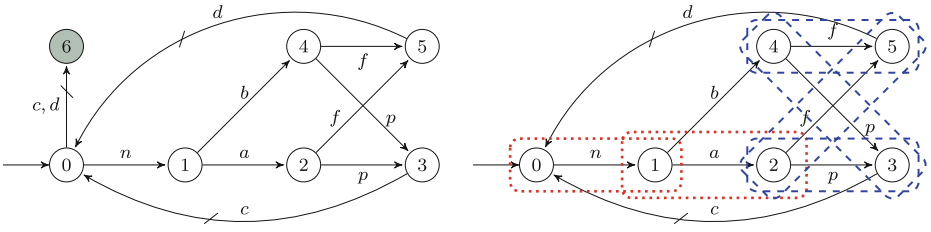
where $M = |\mathcal{A}|$ and the automata also have $O(M)$ states and transitions. This complexity was obtained from a specific example and a generalization was not provided. Section 3 contains the main contribution of this work, a new version of the test for P-observability for an attack set.

3 New test for P-observability for an attack set

This section presents a new test for P-observability for an attack set. Rather than realizing multiple tests of classical observability, the proposed test checks P-observability for an attack set itself. The test is applied over an automaton that implements a desired language K. Before presenting the new test, we show how the attacks can be interpreted visually, through an example and then introduce some definitions.

Example 1 Suppose a conveyor belt transports two types of unfinished products. These products are transported through a test unit, whose automaton representation is shown in Fig. 1a. Whenever a new product arrives (event n), the conveyor belt stops and a sensor identifies the type of product (a or b). A test is then performed according to the type of product. If it passes the test (p), the unfinished product continues its path on the conveyor belt (event c). However, if it fails (f), then the unfinished product must be discarded (d). We want to avoid accepting or discarding a product before performing the test. The uncontrolled behavior of this system is shown in Fig. 1a. State 6, in gray, is a bad state, since it represents the test unit accepting or discarding an unfinished product before testing it.

Now suppose that the test unit has some known vulnerabilities. In one of these vulnerabilities, events n and a are compromised, while in another vulnerability, events f and p are the ones compromised. Thus, we have the following attack set $\mathcal{A} = \{A_{\{a,n\}}, A_{\{f,p\}}\}$. Since in this example there are no unobservable events, attacker $A_\emptyset$ need not be considered. We know that one of these vulnerabilities was exploited by attackers, but we don't know which one. Our goal is to determine if it is possible to obtain a supervisor that will enforce the desired language regardless of which attacker is acting. To accomplish this, we start by considering the automaton which represents the desired language, shown in Fig. 1b. It was obtained by removing from G the bad state. Whenever an attacker acts, it makes an external observer confuse the state the system is in. Consider attacker $A_{\{a,n\}}$. If an observer sees string $y = n$, it cannot be sure if the system is really at state 1 or at state 0 or at state 2. This is because event n may never have happened and was inserted by the attacker or event n happened and was not modified by the attacker or events n and a happened but a was



(a) Automaton G (b) Automaton H

Fig. 1 Automata of Example 1. $\Sigma_c = \{c, d\}$, $\Sigma_u = \{a, b, f, n, p\}$ and $\Sigma_v = \{a, f, n, p\}$

erased by the attacker. Thus, state 0 can be confused with state 1 and state 1 can be confused with state 2, which is represented by the (red) dotted rectangles encircling states 0 and 1 and also 1 and 2. The same reasoning can be applied to find the (blue) dashed rectangles around states 2 and 3, 2 and 5, 4 and 3 and 4 and 5, regarding attacker $A_{\{f,p\}}$. Independent of which attacker is considered, an observer can always confuse a state q with itself. We omit the visual representation for such cases.

An attack can increase the uncertainty for an observer about which state the plant is in, by manipulating the symbols in the observation. This uncertainty can be represented as a relation of pairs of states and each attacker induces a different one. In the previous example, each pair of states in a rectangle is composed of states that can be confused with each other, which is due to the attack or to the partial observation. Thus, each rectangle gives rise to a pair of states in the relation of states that can be confused with each other and with respect to an attacker, where the pair order is given by the transition connecting both states. One can easily conclude that if the pairs (q_1, q) and (q, q_2) are in the relation, for $(q_1, \sigma, q), (q, \sigma', q_2) \in \delta$, then the pair (q_1, q_2) should also be, since the attacker can insert or erase both events. In other words, we can say that this relation is transitive. The following definition formalizes a binary relation on the set of states of an automaton induced by an attacker. Henceforward, this relation, denoted by Π_α, will be called the relation of *indistinguishable* states with respect to attacker $A_\alpha \in \mathcal{A}$.

Definition 1 (Indistinguishable states with respect to attacker A_α) For a given attacker $A_\alpha \in \mathcal{A}$ and automaton G, the relation $\Pi_\alpha \subseteq Q \times Q$ defined as

$$\Pi_\alpha := \{(q, q') \in Q \times Q | (\forall wu \in \mathcal{L}(G))[\delta(q_0, w) = q \wedge \delta(q_0, wu) = q' \wedge u \in (\alpha \cup \Sigma_{uo})^*]\} \tag{8}$$

is the relation of indistinguishable states with respect to attacker A_α.

In words, Π_α has all pairs of states (q, q') such that state q' is reachable from state q with events in α, the events that attacker A_α can manipulate, or with events in Σ_{uo}. This definition was inspired by Wang et al. (2007), where pairs of indistinguishable states arise due only to partial observation. The next example shows the application of Definition 1 over the system considered in the previous example.

Example 2 Let us consider the automaton of Example 1 and also the attack set $\mathcal{A} = \{A_{\{a,n\}}, A_{\{f,p\}}\}$. Using Definition 1, we obtain:

$$\Pi_{\{a,n\}} = \{(0,0), (0,1), (0,2), (1,1), (1,2), (2,2), (3,3), (4,4), (5,5)\};$$
$$\Pi_{\{f,p\}} = \{(0,0), (1,1), (2,2), (2,3), (2,5), (3,3), (4,3), (4,4), (4,5), (5,5)\}.$$

When checking P-observability for an attack set, one operation that is very convenient is to check if two given states $q, q' \in Q$ can be confused with each other. As we do not know which attacker is actually acting, we have to consider the effect of all attackers. This is done by joining all relations of indistinguishable states into one, according to Definition 2.

Definition 2 (Indistinguishable states with respect to attack set $\mathcal{A}$) For an attack set $\mathcal{A}$, the relation $\Pi_\mathcal{A} \subseteq Q \times Q$ defined as

$$\Pi_\mathcal{A} := \bigcup_{A_\alpha \in \mathcal{A}} \Pi_\alpha \tag{9}$$

is the relation of indistinguishable states with respect to attack set $\mathcal{A}$.

The relation $\Pi_{\mathcal{A}}$ is obtained by taking the union of all the pairs in Π_{α}. A pair $(q, q') \in \Pi_{\alpha}$ means that state q' can be reached from state q after the insertion of some events by an attacker $A_{\alpha} \in \mathcal{A}$, making them indistinguishable. Thus, the relation $\Pi_{\mathcal{A}}$ has all pairs of states that are indistinguishable with each other, which is a consequence of the insertion of events by all attackers in the attack set. The next example shows the application of Def. 2.

Example 3 Continuing from Example 2, let us find all pairs of states that can be confused with each other, considering all the attackers in the attack set, using Definition 2.

$$\begin{aligned}\Pi_{\mathcal{A}} &= \Pi_{\{a,n\}} \cup \Pi_{\{f,p\}} \\ &= \{(0,0), (0,1), (0,2), (1,1), (1,2), (2,2), (2,3), (2,5), (3,3), (4,3), \\ &\quad (4,4), (4,5), (5,5)\}.\end{aligned}$$

In order to check P-observability for an attack set, we have to verify if Eq. 6 holds. For any pair of strings in relation $R_{A_{\alpha_i}, A_{\alpha_j}}$, it means that there are two attackers that can take two different strings $w, w' \in K$, respectively, and modify them by inserting and/or erasing events in α_i and α_j, respectively, in a way that results in the same y for both cases. Thus, strings w and y are indistinguishable as well as strings w' and y. An external observer that sees string y cannot decide if the string that actually happened was w or w', i.e., strings w and w' are indistinguishable. Another way to see this is if, hypothetically, we allowed the two attackers to cooperate with each other. Then, one attacker could take string w and change it to y and the second one would take string y and change it to w', making strings w and w' indistinguishable.

In terms of relations of indistinguishable states, we want to know if there is one attacker that can make states $q_1, q \in Q$ indistinguishable and if there is another attacker that can make states $q_2, q \in Q$ also indistinguishable. Thus, if an external observer sees string y such that $\delta(q_0, y) = q$, it cannot be sure if the system is at state q_1 or state q_2. In this case, if we consider, hypothetically, that the attackers are cooperating with each other, then they could make states q_1 and q_2 indistinguishable. Hence, imagining that two attackers can cooperate with each other is useful since it gives us all the states that can be confused with each other. Additionally, if states q_1 and q_2 are a pair of indistinguishable states, then strings w and w', such that $\delta(q_0, w) = q_1$ and $\delta(q_0, w') = q_2$, are also indistinguishable and, therefore, they form a pair in the relation $R_{A_{\alpha_i}, A_{\alpha_j}}$. The effect of considering the attackers cooperating is obtained by composing the relation $\Pi_{\mathcal{A}}$ with itself, as in Definition 3.

Definition 3 (Relation of indistinguishable states for pairwise combined attacks) The relation $\Pi^2_{\mathcal{A}} \subseteq Q \times Q$ defined as

$$\Pi^2_{\mathcal{A}} := (\Pi_{\mathcal{A}} \circ \Pi_{\mathcal{A}}) \cup \widehat{\Pi}_{\mathcal{A}} \tag{10}$$

where $\Pi_{\mathcal{A}}$ is given by Def. 2 and $\widehat{\Pi}_{\mathcal{A}}$ is defined as

$$\widehat{\Pi}_{\mathcal{A}} = \{(q_1, q_2), (q_2, q_1) \in Q \times Q | (\exists q \in Q)[(q, q_1), (q, q_2) \in \Pi_{\mathcal{A}}]\}, \tag{11}$$

is the relation of indistinguishable states for the pairwise combination of attacks.

The next example illustrates the application of Definition 3.

Example 4 Continuing from Example 3, we can use Def. 3 in order to find the effect of all the attackers acting together. From $\Pi_{\mathcal{A}}$ obtained in Example 3,

$$\begin{aligned}
\widehat{\Pi}_{\mathcal{A}} &= \{(0,0),(0,1),(1,0),(0,2),(1,1),(1,2),(2,0),(2,1),(2,2),(2,3),\\
&\quad (2,5),(3,2),(3,3),(3,4),(3,5),\\
&\quad (4,3),(4,4),(4,5),(5,2),(5,3),(5,4),(5,5)\};\\
\Pi_{\mathcal{A}} \circ \Pi_{\mathcal{A}} &= \{(0,0),(0,1),(0,2),(0,3),(0,5),(1,1),(1,2),(1,3),(1,5),(2,2),(2,3),\\
&\quad (2,5),(3,3),(4,3),(4,4),(4,5),(5,5)\}\\
\Pi_{\mathcal{A}}^2 &= \{(0,0),(0,1),(0,2),(0,3),(0,5),(1,0),(1,1),(1,2),(1,3),(1,5),(2,0),\\
&\quad (2,1),(2,2),(2,3),(2,5),(3,2),\\
&\quad (3,3),(3,4),(3,5),(4,3),(4,4),(4,5),(5,2),(5,3),(5,4),(5,5)\}.
\end{aligned}$$

In Example 4, each pair of the relation $\Pi_{\mathcal{A}}^2$ represents a pair of states that are indistinguishable. This means that for each string observed, several possibilities have to be considered. To illustrate, suppose an observer sees string $y = na$. Then, we can consider that:

(i) The system is at state 0, in the case that events n and a never really happened but were inserted by the attacker $\Pi_{\{a,n\}}$;
(ii) The system is at state 1, in the case that only event n happened and a was inserted by attacker $\Pi_{\{a,n\}}$;
(iii) The system is at state 2 if events a and n really happened;
(iv) The system is at state 3 (respectively, 5) if events n, a and p (resp., f) happened but event p (resp., f) was erased by attacker $\Pi_{\{f,p\}}$.

For instance, we can say, from case i), that there exists strings $w = \varepsilon$ and $y = na$ such that $\delta(q_0, w) = 0$ and $y \in A_{\{a,n\}}P(w)$. From case iv), for example, we can say that there exists strings $w' = nap$ and $y = na$ such that $\delta(q_0, w') = 3$ and $y \in A_{\{f,p\}}P(w')$. This allows us to conclude that $A_{\{a,n\}}P(w) \cap A_{\{f,p\}}P(w') \neq \emptyset$. In a more generic way, this means that there are at least two different strings $w, w' \in \mathcal{L}(G)$ and at least two attackers $A_{\alpha_i}, A_{\alpha_j} \in \mathcal{A}$ such that $A_{\alpha_i}P(w) \cap A_{\alpha_j}P(w') \neq \emptyset$. This is true whenever it is possible to reach a state from another by inserting vulnerable events (i.e., following a forward path along the transitions) or if there is a state from which it is possible to reach two different states by inserting vulnerable events and there is at most one change between rectangles that represent different attackers.

The requirement of P-observability for an attack set is only relevant when dealing with strings $w, w' \in K$ that can generate the same observation after the attack and such that there exists an event $\sigma \in \Sigma$ such that $w\sigma \in K$ and $w'\sigma \in \mathcal{L}(G)\backslash K$ or $w'\sigma \in K$ and $w\sigma \in \mathcal{L}(G)\backslash K$. When $w\sigma \in \mathcal{L}(G) \setminus K$ or $w'\sigma \in \mathcal{L}(G) \setminus K$, it means that event σ should be disabled after w or w', respectively. In the same way, when $w\sigma \in K$ or $w'\sigma \in K$, it means that event σ should be enabled after w or w', respectively. Let $\xi : Q \to 2^{\Sigma}$ be a map defined as

$$\xi(q) := \{\sigma \in \Sigma | (\exists w \in \mathcal{L}(G))[\delta(q_0, w) = q \wedge w\sigma \in K]\} \tag{12}$$

that gives, for a state $q \in Q$, the set of *enabled* events at state q. Also, let the map $\phi : Q \to 2^{\Sigma}$ defined as

$$\phi(q) := \{\sigma \in \Sigma | (\exists w \in \mathcal{L}(G))[\delta(q_0, w) = q \wedge w\sigma \in \mathcal{L}(G) \setminus K]\} \tag{13}$$

be the map that gives, for a given state $q \in Q$, the set of *disabled* events at state q. Inspired by Su and Wonham (2004), where the authors present a binary relation of pairs of states

that are consistent with respect to their control action and to their marking, we present the following definition.

Definition 4 (Relation of control-inconsistent states) The binary relation $\mathcal{I} \subseteq Q \times Q$ defined by

$$\mathcal{I} := \{(q, q') \in Q \times Q | \xi(q) \cap \phi(q') \neq \emptyset \vee \xi(q') \cap \phi(q) \neq \emptyset\} \tag{14}$$

is the relation of control-inconsistent states.

In words, according to Definition 4, a pair of states is in the relation of control inconsistent states $\mathcal{I}$ if the control action at these two states is conflicting. It is important to notice that the relation $\mathcal{I}$ is not transitive but is symmetric. For testing P-observability for an attack set, it is not necessary to consider all pairs of strings $w, w' \in K$, but only the pairs such that their control action is conflicting. In other words, we need to consider only the states that are control inconsistent. Next, we present the main result of this work, a new test for P-observability for an attack set.

Theorem 2 *Let G be a deterministic finite automaton that represents the behavior of a system, K the desired language, with $K \subseteq \mathcal{L}(G)$. The set $\Sigma_{uo} \subseteq \Sigma$ is the set of unobservable events, $\Sigma_v \subseteq \Sigma$ is the set of vulnerable events ($\Sigma_{uo} \cap \Sigma_v = \emptyset$) and $\mathcal{A} = \{A_{\alpha_1}, A_{\alpha_2}, \ldots, A_{\alpha_M}\}$ is the attack set, with $\alpha_i \subseteq \Sigma_v$, $i = 1, \ldots, M$. Let $\mathcal{I}$ be the relation of control inconsistent states of the automaton that implements K. The language K will be P-observable for $\mathcal{A}$ and $\mathcal{L}(G)$ if and only if $\Pi^2_{\mathcal{A}} \cap \mathcal{I} = \emptyset$.*

Theorem 2 states that a given desired language K is P-observable for an attack set $\mathcal{A}$ if and only if there are no mutual pairs of states between the relations $\Pi^2_{\mathcal{A}}$ and $\mathcal{I}$. One can apply Theorem 2 to verify if a language K is P-observable for an attack set $\mathcal{A}$ by checking if $(q, q') \notin \Pi^2_{\mathcal{A}}$ holds for every pair (q, q') in the relation of control inconsistent states $\mathcal{I}$.

Before presenting the proof of Theorem 2, we present Lemma 1, that is used in its proof.

Lemma 1 *Let $G = (Q, \Sigma, \delta, q_0)$ be an automaton, $q, q' \in Q$ be states such that $(q, q') \in \Pi_\alpha$ for some attacker $A_\alpha \in \mathcal{A}$ and $P : \Sigma^* \to \Sigma^*_{uo}$ be a natural projection. Then, for all strings $w \in \mathcal{L}(G)$ and $v \in w(\alpha \cup \Sigma_{uo})^*$, such that $\delta(q_0, w) = q$ and $\delta(q_0, v) = q'$, it holds that*

1) $v \in P^{-1}(A_\alpha P(w))$;
2) $w \in P^{-1}(A_\alpha P(v))$.

Proof of Lemma 1 Since $v \in w(\alpha \cup \Sigma_{uo})^*$, to show that $v \in P^{-1}(A_\alpha P(w))$, it suffices to show that $w(\alpha \cup \Sigma_{uo})^* \subseteq P^{-1}(A_\alpha P(w))$. We start by writing w as $w = \sigma_1\sigma_2 \ldots \sigma_n$, $\sigma_i \in \Sigma$, for $i = 1, 2, \ldots, n$. Then $P(w) = P(\sigma_1)P(\sigma_2) \ldots P(\sigma_n)$. To obtain $A_\alpha(P(\sigma_1) \ldots P(\sigma_n))$, we first remove all events in α from $P(\sigma_1) \ldots P(\sigma_n)$, resulting in $R_{\neg\alpha}(P(\sigma_1)) \ldots R_{\neg\alpha}(P(\sigma_n))$. Then we find all strings that are equal to $R_{\neg\alpha}(P(\sigma_1)) \ldots R_{\neg\alpha}(P(\sigma_n))$ if the events in α were also removed, i.e., $\alpha^* R_{\neg\alpha}(P(\sigma_1))\alpha^* \ldots \alpha^* R_{\neg\alpha}(P(\sigma_n))\alpha^*$. Thus,

$$\begin{aligned} P^{-1}(A_\alpha P(w)) &= P^{-1}(A_\alpha(P(w))) \\ &= P^{-1}(A_\alpha(P(\sigma_1)P(\sigma_2) \ldots P(\sigma_n))) \\ &= P^{-1}(\alpha^* R_{\neg\alpha}(P(\sigma_1))\alpha^* \ldots \alpha^* R_{\neg\alpha}(P(\sigma_n))\alpha^*). \end{aligned} \tag{15}$$

In addition, $P(w)\alpha^* = P(\sigma_1)P(\sigma_2)\dots P(\sigma_n)\alpha^* \subseteq \alpha^* R_{\neg\alpha}(P(\sigma_1))\alpha^* \dots \alpha^* R_{\neg\alpha}(P(\sigma_n))\alpha^*$. Then we can say that

$$\begin{aligned} P(w)\alpha^* &\subseteq \alpha^* R_{\neg\alpha}(P(\sigma_1))\alpha^* \dots \alpha^* R_{\neg\alpha}(P(\sigma_n))\alpha^* \\ P^{-1}(P(w)\alpha^*) &\subseteq P^{-1}(\alpha^* R_{\neg\alpha}(P(\sigma_1))\alpha^* \dots \alpha^* R_{\neg\alpha}(P(\sigma_n))\alpha^*). \end{aligned} \tag{16}$$

Taking the left side of Eq. 16,

$$P^{-1}(P(w)\alpha^*) = P^{-1}(P(w))P^{-1}(\alpha^*) \tag{17}$$

and as $w \in P^{-1}P(w)$ and $(\alpha \cup \Sigma_{uo})^* \subseteq P^{-1}(\alpha^*)$, then

$$w(\alpha \cup \Sigma_{uo})^* \subseteq P^{-1}(P(w))P^{-1}(\alpha^*). \tag{18}$$

As $v \in w(\alpha \cup \Sigma_{uo})^*$, then it follows from Eq. 18 that $v \in P^{-1}(P(w))P^{-1}(\alpha^*)$. From Eq. 17, $v \in P^{-1}(P(w)\alpha^*)$ and, from Eq. 16, $v \in P^{-1}(\alpha^* R_{\neg\alpha}(P(\sigma_1))\alpha^* \dots \alpha^* R_{\neg\alpha}(P(\sigma_n))\alpha^*)$. Finally, Eq. 15 allows us to say that $v \in P^{-1}(A_\alpha P(w))$, proving 1).

For the second part, if $v \in w(\alpha \cup \Sigma_{uo})^*$ and $v \in \mathcal{L}(G)$, then $v = wu$, for all $u \in (\alpha \cup \Sigma_{uo})^*$ such that $wu \in \mathcal{L}(G)$. The strings w and u can be written as $w = \sigma_1\sigma_2\dots\sigma_n$, with $\sigma_i \in \Sigma$, for $i = 1, 2, \dots n$ and $u = \gamma_1\gamma_2\dots\gamma_m$, with $\gamma_j \in (\alpha \cup \Sigma_{uo})$, for $j = 1, 2, \dots m$. Thus, v can be written as $v = \sigma_1\sigma_2\dots\sigma_n\gamma_1\gamma_2\dots\gamma_m$. Applying the natural projection, we have

$$\begin{aligned} P(v) &= P(\sigma_1\sigma_2\dots\sigma_n\gamma_1\gamma_2\dots\gamma_m) \\ &= P(\sigma_1)P(\sigma_2)\dots P(\sigma_n)P(\gamma_1)P(\gamma_2)\dots P(\gamma_m). \end{aligned} \tag{19}$$

Now, we apply the map A_α over (19):

$$A_\alpha(P(v)) = A_\alpha(P(\sigma_1)\dots P(\sigma_n)P(\gamma_1)\dots P(\gamma_m)). \tag{20}$$

To obtain $A_\alpha(P(\sigma_1)\dots P(\sigma_n)P(\gamma_1)\dots P(\gamma_m))$, we first remove all events in α from $P(\sigma_1)\dots P(\sigma_n)P(\gamma_1)\dots P(\gamma_m)$, resulting in $R_{\neg\alpha}(P(\sigma_1))\dots R_{\neg\alpha}(P(\sigma_n))R_{\neg\alpha}(P(\gamma_1))\dots R_{\neg\alpha}(P(\gamma_m))$. For each $P(\gamma_j)$, there are two possibilities: *i)*$\gamma_j \in \Sigma_{uo}$ and $P(\gamma_j) = \varepsilon$ and; *ii)*$\gamma_j \in \alpha$ and $P(\gamma_j) = \gamma_j \in \alpha$. Either way, if the events in α are removed, it results in $R_{\neg\alpha}(P(\sigma_1))\dots R_{\neg\alpha}(P(\sigma_n))$. Then we find all strings that are equal to $R_{\neg\alpha}(P(\sigma_1))\dots R_{\neg\alpha}(P(\sigma_n))$ if the events in α were also removed, i.e., $\alpha^* R_{\neg\alpha}(P(\sigma_1))\alpha^*\dots\alpha^* R_{\neg\alpha}(P(\sigma_n))\alpha^*$. Thus, we can write (20) as

$$A_\alpha(P(v)) = \alpha^* R_{\neg\alpha}(P(\sigma_1))\alpha^*\dots\alpha^* R_{\neg\alpha}(P(\sigma_n))\alpha^*. \tag{21}$$

As $P(w) = P(\sigma_1)\dots P(\sigma_n)$, then $P(w)$ is an element of $\alpha^* R_{\neg\alpha}(P(\sigma_1))\alpha^*\dots\alpha^* R_{\neg\alpha}(P(\sigma_n))\alpha^*$, which allows us to say that $P(w) \in A_\alpha(P(v)) = A_\alpha P(v)$. Applying the inverse projection on both sides, $P^{-1}(P(w)) \subseteq P^{-1}(A_\alpha P(v))$. Also, $w \in P^{-1}(P(w))$, allowing us to say that $w \in P^{-1}(A_\alpha P(v))$. This proves 2). □

Lemma 1 can be interpreted as follows. If $(q, q') \in \Pi_\alpha$, for some attacker $A_\alpha \in \mathcal{A}$, $q, q' \in Q$ and strings $v, w \in \mathcal{L}(G)$ such that $\delta(q_0, w) = q$ and $\delta(q_0, v) = q'$, then $v \in P^{-1}(A_\alpha P(w))$ means that v is obtained from w after the insertion of events in α by attacker A_α and/or after the occurrence of unobservable events, while $w \in P^{-1}(A_\alpha P(v))$ means that w is obtained from v after the deletion of events in α by attacker A_α and/or after the insertion of unobservable events. Now we can present the theorem's proof.

Proof of Theorem 2 ($\Rightarrow$) We proceed by contradiction. Suppose K is P-observable and $\Pi^2_{\mathcal{A}} \cap \mathcal{I} \neq \emptyset$. Let $q_1, q_2 \in Q$ be states such that $(q_1, q_2) \in \Pi^2_{\mathcal{A}} \cap \mathcal{I}$ and $w, w' \in \mathcal{L}(G)$

be strings such that $\delta(q_0, w) = q_1$ and $\delta(q_0, w') = q_2$. If $(q_1, q_2) \in \mathcal{I}$, then $\exists \sigma \in \Sigma$ such that $w\sigma \in K$ and $w'\sigma \in \mathcal{L}(G) \setminus K$ or $w\sigma \in \mathcal{L}(G) \setminus K$ and $w'\sigma \in K$. This means that $(w, w') \notin \text{act}_{K \subset \mathcal{L}(G)}$. If $(q_1, q_2) \in \Pi^2_{\mathcal{A}}$, three cases can happen:

1. (q_1, q_2) existed already in $\Pi_{\mathcal{A}}$. In such case $(q_1, q_2) \in \Pi_{\alpha_i}$, for some attacker $A_{\alpha_i} \in \mathcal{A}$. Using Lemma 1, we have that $w \in P^{-1}(A_{\alpha_i} P(w'))$. Because it is always true that $w \in P^{-1}(A_{\alpha_i} P(w))$, we can conclude that $P^{-1}(A_{\alpha_i} P(w)) \cap P^{-1}(A_{\alpha_i} P(w')) = P^{-1}(A_{\alpha_i} P(w) \cap A_{\alpha_i} P(w')) \neq \emptyset$. Then, it is possible to say that $(w, w') \in R_{A_{\alpha_i}, A_{\alpha_j}}$, $A_{\alpha_i} = A_{\alpha_j}$.
2. (q_1, q_2) appeared as a result of $\Pi_{\mathcal{A}} \circ \Pi_{\mathcal{A}}$. In such a case, $\exists q \in Q, q \neq q_2, q \neq q_1$ such that $(q_1, q), (q, q_2) \in \Pi_{\mathcal{A}}$. Without loss of generality, we can say that $(q_1, q) \in \Pi_{\alpha_i}$ and $(q, q_2) \in \Pi_{\alpha_j}$ for attackers $A_{\alpha_i}, A_{\alpha_j} \in \mathcal{A}$ and $\exists v \in \mathcal{L}(G)$ such that $w \leq v \leq w'$, $\delta(q_0, v) = q$. According to Definition 1, $v \in w(\alpha_i \cup \Sigma_{uo})^*$ and $w' \in v(\alpha_j \cup \Sigma_{uo})^*$. Thus, using Lemma 1, we can say that $v \in P^{-1}(A_{\alpha_i} P(w))$ and $v \in P^{-1}(A_{\alpha_j} P(w'))$, which implies that $v \in P^{-1}(A_{\alpha_i} P(w)) \cap P^{-1}(A_{\alpha_j} P(w')) \neq \emptyset$.
3. (q_1, q_2) appeared as a result of $\widehat{\Pi}_{\mathcal{A}}$. In such a case, there exists $q \in Q$, such that $(q, q_1), (q, q_2) \in \Pi_{\mathcal{A}}$. Without loss of generality, we can say that $(q, q_1) \in \Pi_{\alpha_i}$ and $(q, q_2) \in \Pi_{\alpha_j}$ for attackers $A_{\alpha_i}, A_{\alpha_j} \in \mathcal{A}$ and $\exists v \in \mathcal{L}(G)$ such that $v \leq w$ and $v \leq w'$, $\delta(q_0, v) = q$. According to Definition 1, $w \in v(\alpha_i \cup \Sigma_{uo})^*$ and $w' \in v(\alpha_j \cup \Sigma_{uo})^*$. Thus, using Lemma 1, we can say that $v \in P^{-1}(A_{\alpha_i} P(w))$ and $v \in P^{-1}(A_{\alpha_j} P(w'))$, which implies that $v \in P^{-1}(A_{\alpha_i} P(w)) \cap P^{-1}(A_{\alpha_j} P(w')) \neq \emptyset$.

In all of the presented cases, we conclude that $P^{-1}(A_{\alpha_i} P(w)) \cap P^{-1}(A_{\alpha_j} P(w')) = P^{-1}(A_{\alpha_i} P(w) \cap A_{\alpha_j} P(w')) \neq \emptyset$, which implies that $A_{\alpha_i} P(w) \cap A_{\alpha_j} P(w') \neq \emptyset$. Thus, whenever $(q_1, q_2) \in \Pi^2_{\mathcal{A}}$, it is possible to say that $(w, w') \in R_{A_{\alpha_i}, A_{\alpha_j}}$. Hence, $(w, w') \in R_{A_{\alpha_i}, A_{\alpha_j}}$ but $(w, w') \notin \text{act}_{K \subset L}$, which contradicts the definition of P-observability for an attack set.

($\Leftarrow$) Again, by contradiction, suppose that K is not P-observable and $\Pi^2_{\mathcal{A}} \cap \mathcal{I} = \emptyset$. If K is not P-observable, it means that $\exists w, w' \in \Sigma^*$ and $\exists A_{\alpha_i}, A_{\alpha_j} \in \mathcal{A}$ such that $A_{\alpha_i} P(w) \cap A_{\alpha_j} P(w') \neq \emptyset$ ($(w, w') \in R_{A_{\alpha_i}, A_{\alpha_j}}$) and also $\exists \sigma \in \Sigma$ such that $w\sigma \in K$ and $w'\sigma \in \mathcal{L}(G) \setminus K$ or $w\sigma \in \mathcal{L}(G) \setminus K$ and $w'\sigma \in K$ ($(w, w') \notin \text{act}_{K \subset \mathcal{L}(G)}$).

Without loss of generality, let w and w' be strings such that $\delta(q_0, w) = q_1$, $\delta(q_0, w') = q_2$, with $(q_1, q_2) \in \mathcal{I}$. Furthermore, as $(w, w') \in R_{A_{\alpha_i}, A_{\alpha_j}}$, then $A_{\alpha_i} P(w) \cap A_{\alpha_j} P(w') \neq \emptyset$ and $P^{-1}(A_{\alpha_i} P(w) \cap A_{\alpha_j} P(w')) \neq \emptyset$, for some $A_{\alpha_i}, A_{\alpha_j} \in \mathcal{A}$. Let v be a string such that $v \in P^{-1}(A_{\alpha_i} P(w) \cap A_{\alpha_j} P(w')) \cap \mathcal{L}(G)$ and $q \in Q$ be a state such that $\delta(q_0, v) = q$. Then, $v \in P^{-1}(A_{\alpha_i} P(w))$ and $v \in P^{-1}(A_{\alpha_j} P(w'))$, i.e., v is an attacked version of w and an attacked version of w'. Several cases can be considered regarding the relationship between v, w and w':

1. $w \leq v$ and $v \leq w'$;
2. $w' \leq v$ and $v \leq w$;
3. $w \leq v$ and $w' \leq v$;
4. $v \leq w$ and $v \leq w'$.

For case 1, $\exists u \in (\alpha_i \cup \Sigma_{uo})^*$ and $\exists u' \in (\alpha_j \cup \Sigma_{uo})^*$ such that $wu = v$ and $vu' = w'$. Using Def. 1, we can say that $(q_1, q) \in \Pi_{\alpha_i}$ and $(q, q_2) \in \Pi_{\alpha_j}$. Then, $(q_1, q), (q, q_2) \in \Pi_{\mathcal{A}}$ and consequently, $(q_1, q_2) \in \Pi^2_{\mathcal{A}}$.

Similarly, for case 2, $\exists u \in (\alpha_i \cup \Sigma_{uo})^*$ and $\exists u' \in (\alpha_j \cup \Sigma_{uo})^*$ such that $w'u = v$ and $vu' = w$. Using again Def. 1, we can say that $(q_2, q) \in \Pi_{\alpha_i}$ and $(q, q_1) \in \Pi_{\alpha_j}$. Then, $(q_2, q), (q, q_1) \in \Pi_{\mathcal{A}}$ and consequently, $(q_2, q_1) \in \Pi^2_{\mathcal{A}}$.

For case 3, $w \leq v$ and $w' \leq v$ imply that $w \leq w'$ or $w' \leq w$ since G is deterministic. If $w \leq w'$, then $\exists u \in (\alpha_i \cup \Sigma_{uo})^*$ such that $wu = w'$ and all of the following are true: $(q_1, q_2) \in \Pi_{\alpha_i}$, $(q_1, q_2) \in \Pi_{\mathcal{A}}$ and finally $(q_1, q_2) \in \Pi^2_{\mathcal{A}}$. Similarly, if $w' \leq w$, $\exists u \in (\alpha_j \cup \Sigma_{uo})^*$ such that $w'u = w$ and $(q_2, q_1) \in \Pi_{\alpha_j}$, $(q_2, q_1), \in \Pi_{\mathcal{A}}$ and finally $(q_2, q_1) \in \Pi^2_{\mathcal{A}}$.

For the last case, two new cases are considered:

- $w \leq w'$ or $w' \leq w$. This is identical to case 3.
- v is a prefix of w and v is a prefix of w' but none of them is a prefix of the other. Then, $\exists u \in (\alpha_i \cup \Sigma_{uo})^*$ and $\exists u' \in (\alpha_j \cup \Sigma_{uo})^*$ such that $vu = w$ and $vu' = w'$. From Def. 1, $(q, q_1) \in \Pi_{\alpha_i}$ and $(q, q_2) \in \Pi_{\alpha_j}$ for attackers $A_{\alpha_i}, A_{\alpha_j} \in \mathcal{A}$. Then, $(q, q_1), (q, q_2) \in \Pi_{\mathcal{A}}$, $(q_1, q_2), (q_2, q_1) \in \widehat{\Pi}_{\mathcal{A}}$ and consequently, $(q_1, q_2), (q_2, q_1) \in \Pi^2_{\mathcal{A}}$.

Since $\mathcal{I}$ is symmetric, i.e., $(q_1, q_2), (q_2, q_1) \in \mathcal{I}$, all the presented cases allow us to conclude that $\Pi^2_{\mathcal{A}} \cap \mathcal{I} \neq \emptyset$, contradicting the initial hypothesis. Therefore, we can say that K is P-observable if and only if $\Pi^2_{\mathcal{A}} \cap \mathcal{I} = \emptyset$. □

The next example uses Theorem 2 to test if a given language is P-observable for an attack set.

Example 5 From Example 1, Fig. 1b represents the desired language of our system. By comparing Fig. 1a and b, we can conclude that events c and d should be disabled at state 0, while they can occur at states 3 and 5, respectively. Thus, applying Def. 4, we obtain: $\mathcal{I} = \{(0, 3), (0, 5), (3, 0), (5, 0)\}$. From Example 4, we can say that $(0, 3) \in \Pi^2_{\mathcal{A}}$. Then, it is true that $(0, 3) \in \mathcal{I}$, $(0, 3) \in \Pi^2_{\mathcal{A}}$ and $\Pi^2_{\mathcal{A}} \cap \mathcal{I} \neq \emptyset$. Hence, using Theorem 2 we can conclude that K is not P-observable for the attack set $\mathcal{A}$.

The next section introduces algorithms that allows us to perform the P-observability test in a computational tool.

4 Algorithms

In this section we present the algorithms developed to perform the P-observability test. We start by presenting Algorithm 1, shown in Fig. 2a, that obtains the relation of indistinguishable states Π_α with respect to attacker A_α. The algorithm receives an automaton $H = (Q^H, \Sigma, \delta^H, q_0^H)$, such that $\mathcal{L}(H) = K$, and the set α of events that attacker $A_\alpha \in \mathcal{A}$ can manipulate.

In Alg. 1, Ξ represents a FIFO queue and the call to the function at line 5 sets the attribute of the states, which is the number of outgoing transitions from each state. This attribute is represented by $q.out$, for $q \in Q^H$. The operation at line 1 applies the map defined in Eq. 4 over all transitions in δ, as follows:

$$R_{\neg(\Sigma \setminus \alpha)}(\delta^H) := \{(q, \sigma, q') \in \delta^H | \sigma \in \alpha\}. \tag{22}$$

In words, the operation $R_{\neg(\Sigma \setminus \alpha)}(\delta)$ removes from δ all the transitions labeled with non-vulnerable events, i.e., events in $\Sigma \setminus \alpha$, resulting in δ_α. The idea behind Alg. 1 is to first obtain a modified automaton $H' = (Q, \Sigma, \delta_\alpha, q_0)$ and for each state $q \in Q$, a breadth-first search (BFS) is performed. Once state $q' \in Q$ is reached from q, the pair (q, q') is added to

Algorithm 1. Indistinguishable states Π_α with respect to attacker A_α
Input: $H = (Q^H, \Sigma, \delta^H, q_0^H), \alpha$
Output: Π_α
$\delta_\alpha \leftarrow R_{\neg(\Sigma\setminus\alpha)}(\delta^H)$
$\Xi \leftarrow \emptyset$ // Queue
$\pi \leftarrow \emptyset$
$\Pi \leftarrow \emptyset$
INITIALIZE_STATE_ATTRIBUTES()
for all $q \in Q^H$ **do**
  Ξ.ENQUEUE(q)
  while $\Xi \neq \emptyset$ **do**
    $q_c \leftarrow \Xi$.DEQUEUE()
    if $(q, q_c) \notin \Pi$ **then**
      $\Pi \leftarrow \Pi \cup \{(q, q_c)\}$
    end if
    if $q_c \notin \pi$ **then**
      $\pi \leftarrow \pi \cup \{q_c\}$
    end if
    if $q_c.out > 0$ **then**
      for all $(q_c, \sigma, q') \in \delta_\alpha$ **do**
        if $q' \notin \pi$ **then**
          Ξ.ENQUEUE(q')
        end if
      end for
    end if
  end while
  $\pi \leftarrow \emptyset$
end for
return Π_α

(a)

Algorithm 2. P-observability test
Input: $H = (Q^H, \Sigma, \delta^H, q_0^H), \mathcal{I}, \mathcal{A}$
Output: {TRUE, FALSE}
$\Pi_\mathcal{A} \leftarrow \emptyset$
for all $A_\alpha \in \mathcal{A}$ **do**
  $\Pi_\alpha \leftarrow$ INDISTINGUISHABLE_STATES(H, α)
  $\Pi_\mathcal{A} \leftarrow \Pi_\mathcal{A} \cup \Pi_\alpha$
end for
$\Pi^2_\mathcal{A} \leftarrow (\Pi_\mathcal{A} \circ \Pi_\mathcal{A}) \cup \widehat{\Pi}_A$
for all $(q_1, q_2) \in \mathcal{I}$ **do**
  if $(q_1, q_2) \in \Pi^2_\mathcal{A}$ **then**
    return FALSE
  end if
end for
return TRUE

(b)

Algorithm 3. Inconsistent states
Input: $G = (Q^G, \Sigma, \delta^G, q_0^G)$, $H = (Q^H, \Sigma, \delta^H, q_0^H)$
Output: $\mathcal{I}$
$\mathcal{I} \leftarrow \emptyset$
for all $q \in Q^H$ **do**
  for all $q' \in Q^H \setminus \{q\}$ **do**
    if $\phi(q) \cap \xi(q') \neq \emptyset \vee \phi(q') \cap \xi(q) \neq \emptyset$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{(q, q')\}$
    end if
  end for
end for
return $\mathcal{I}$

(c)

Fig. 2 Algorithms

Π_α. The set π keeps track of all states already visited in a particular BFS, guaranteeing that each state is visited only once.

Algorithm 2, shown in Fig. 2b, is used to check if a given desired language is P-observable for an attack set. It receives an automaton H such that $\mathcal{L}(H) = K$, the relation of pairs of control inconsistent states $\mathcal{I}$ and the attack set $\mathcal{A}$. It returns TRUE if K is P-observable or FALSE, otherwise.

In Alg. 2 we assume that the relation of control inconsistent states is already provided. Algorithm 3, shown in Fig. 2c, shows how $\mathcal{I}$ can be obtained. The algorithm receives two automata $G = (Q^G, \Sigma, \delta^G, q_0^G)$ and $H = (Q^H, \Sigma, \delta^H, q_0^H)$, representing the plant and the desired language, respectively. We assume that $Q^H \subseteq Q^G$. To obtain $\xi(q)$, for $q \in Q^H$, we get all $\sigma \in \Sigma$ such that $(q, \sigma, q') \in \delta^H$, for some $q' \in Q^H$. On the other hand, $\phi(q)$, for $q \in Q^H$, is obtained by picking all $\sigma \in \Sigma$ such that $(q, \sigma, q') \in \delta^G$ and $(q, \sigma, q') \notin \delta^H$, for some $q' \in Q^G$. Then, for all pairs of states $q, q' \in Q^H$, it is verified if $\phi(q) \cap \xi(q') \neq \emptyset$ or $\phi(q') \cap \xi(q) \neq \emptyset$.

The time complexity of Alg. 1 is $O(|Q^G||\delta^G|)$ while for Alg. 2 it is $O(|\mathcal{A}||Q^G||\delta^G| + |\mathcal{A}||Q^G|^2 + |Q^G|^3)$. The time complexity for obtaining the relation of control inconsistent states $\mathcal{I}$ is $O(|Q^G|^2(|\Sigma| + |\delta^G|^2))$.

5 Conclusion

As the use of CPSs is increasing, carrying with them a voluminous use of communication networks, the surface area exposed to malicious agents is growing as well. This justifies the endeavor of finding solutions to reliable communication. Within the supervisory control

context, one of these solutions is to design resilient supervisors, that can guarantee the legal behavior of the system, regardless of the attack.

For attacks in the output symbols, one of the conditions that allows robust supervisors to be designed is the P-observability for an attack set. Although a test for this property already existed, it was based on a series of tests of the classical observability property. Our contributions are the following: we introduce a new test that checks for P-observability for an attack set itself and considers the effect of all attackers in a single run; we provide a visual interpretation that makes it easy to see at a glance which states are confusable with other states for each potential attack; and our development produces new concepts which yield explanations for how an attacker acts, thus providing some insight into the attacks under consideration. In addition, we presented an algorithm for checking P-observability for an attack set, that directly applied the definitions proposed and whose overall complexity is $O(|\mathcal{A}||Q||\delta| + |\mathcal{A}||Q|^2 + |Q|^3)$.

Although there is a gap between our attack model and its application in a real world problem, our approach provides a better understanding about a recently described property, whose application is still incipient in the literature. Thus, we open new fronts for the research of more realistic models. We are currently investigating the effect of considering multiple attackers acting at once, as well as the effect of having them acting on events from the supervisor to the plant.

Acknowledgments This work has been supported by the National Council for Scientific and Technological Development - CNPq under grant 443656/2018-5, CAPES, Brazil, Fapemig and PRPq-UFMG and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Carvalho LK, Wu YC, Kwong R, Lafortune S (2016) Detection and prevention of actuator enablement attacks in supervisory control systems. 13th International Workshop on Discrete Event Systems, pp 298–305

Carvalho LK, Wu YC, Kwong R, Lafortune S (2018) Detection and mitigation of classes of attacks in supervisory control systems. Automatica 97:121–133

Cassandras C, Lafortune S (2007) Introduction to Discrete Event Systems, 2nd edn. Springer, New York

Fritz R, Zhang P (2018) Modeling and detection of cyber attacks on discrete event systems. IFAC-PapersOnLine 51(7):285–290

Gao C, Seatzu C, Li Z, Giua A (2019) Multiple attacks detection on discrete event systems. IEEE International Conference on Systems, Man and Cybernetics, pp 2352–2357

Ji Y, Yin X, Lafortune S (2019) Opacity enforcement using nondeterministic publicly known edit functions. IEEE Trans Autom Control 64(10):4369–4376

Li Y, Tong Y, Giua A (2020) Detection and prevention of cyber attacks in networked control systems. In: Proceedings of the 15th IFAC Workshop on Discrete Event Systems, pp 7–13

Lima PM, Alves MVS, Carvalho LK, Moreira MV (2019) Security against communication network attacks of cyber-physical systems. J Control Autom Electr Syst 30(1):125–135

Lin L, Su R (2020) Synthesis of covert actuator and sensor attackers as supervisor synthesis. In: Proceedings of the 15th IFAC Workshop on Discrete Event Systems, vol 1-6, pp 561–577

Lin L, Zhu Y, Su R (2019) Towards bounded synthesis of resilient supervisors against actuator attacks. In: Proceedings of the 58th IEEE Conference on Decision and Control. IEEE, pp 7659–7664

Lu Y (2017) Cyber physical system (CPS)-based industry 4.0: A Survey. J Ind Integr Manag 02(03):17500141–175001457

Meira-Góes R, Kang E, Kwong RH, Lafortune S (2020) Synthesis of sensor deception attacks at the supervisory layer of Cyber–Physical systems. Automatica 121:109172

Meira-Góes R, Lafortune S, Marchand H (2021) Synthesis of supervisors robust against sensor deception attacks. IEEE Trans Autom Control 9286(c):1–8

Mohajerani S, Meira-Góes R, Lafortune S (2020) Efficient synthesis of sensor deception attacks using observation Equivalence-Based abstraction. In: Proceedings of the 15th IFAC Workshop on Discrete Event Systems, pp 28–34

Rashidinejad A, Lin L, Wetzels B, Zhu Y, Reniers M, Su R (2019) Supervisory control of discrete-event systems under attacks: an overview and outlook. In: Proceedings of the 18th European Control Conference. EUCA, pp 1732–1739

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

Su R, Wonham WM (2004) Supervisor reduction for discrete-event systems. Discret Event Dyn Syst 14:31–53

Wakaiki M, Tabuada P, Hespanha JP (2019) Supervisory control of discrete-event systems under attacks. Dyn Games Appl 9(4):965–983

Wang W, Lafortune S, Lin F (2007) An algorithm for calculating indistinguishable states and clusters in Finite-State automata with partially observable transitions. Syst Control Lett 56(9-10):656–661

Wang Y, Pajic M (2019) Supervisory control of discrete event systems in the presence of sensor and actuator attacks. In: Proceedings of the 58th IEEE Conference on Decision and Control. IEEE, pp 5350–5355

Zhang Q, Li Z, Seatzu C, Giua A (2018) Stealthy attacks for partially-observed discrete event systems. IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp 1161–1164

Zhang Q, Seatzu C, Li Z, Giua A (2021) State estimation under attack in partially-observed discrete event systems. arXiv:1906.10207v3