IMPLEMENTATION OF A SECURITY MODULE FOR CYBER-PHYSICAL SYSTEMS

Públio M. Lima*, Lilian K. Carvalho*, Marcos V. S. Alves*, Marcos V. Moreira*

*COPPE - Electrical Engineering Program, Universidade Federal do Rio de Janeiro, Cidade Universitária, Ilha do Fundão, Rio de Janeiro, 21.945-970, RJ, Brazil.

Emails: publio@poli.ufrj.br, lilian.carvalho@poli.ufrj.br, mvalves@poli.ufrj.br, moreira.mv@poli.ufrj.br

Abstract— Communication networks are commonly used to connect sensors, actuators, and controllers to monitor and control Cyber-Physical Systems (CPS). The use of communication networks increases the vulnerability of the CPS to cyber attacks that can drive the system to unsafe states. One of the most powerful cyber attacks is the so-called man-in-the-middle attack, where the intruder can observe, hide, create or replace information in the attacked network channel. In a previous paper we introduced the definition of NA-Safe Controllability, that is related with the capability of detecting intrusions and preventing damages caused by man-in-the-middle attacks in the sensor and/or control communication channels in supervisory control systems. We present in this paper a method for the implementation of a security module based on the definition of NA-Safe Controllability proposed in our previous work.

Keywords— Cyber-Physical Systems, Security, Cyber Attacks, Discrete-Event Systems, Automata.

Resumo— Redes de comunicação são frequentemente usadas para conectar sensores, atuadores e controladores em sistemas ciber-físicos (SCF). O uso de comunicação por redes aumenta a vulnerabilidade do sistema a ataques cibernéticos. Um dos ataques cibernéticos mais poderosos é o denominado *man-in-the-middle*, em que o intruso pode ler, esconder, criar ou modificar a informação que flui no canal de rede atacado. Em um artigo anterior é introduzida a definição de *NA-Safe Controllability*, que é relacionada com a capacidade de detectar intrusos e prevenir danos causados por ataques do tipo *man-in-the-middle* nos canais de comunicação de sensores e/ou atuadores em sistemas de controle supervisório. Neste artigo é apresentado um método para a implementação de um módulo de segurança baseado na definição de *NA-Safe Controllability* proposta em nosso trabalho anterior.

Palavras-chave— Sistemas Ciber-Físicos, Segurança, Ataques Cibernéticos, Sistemas a Eventos Discretos, Autômatos.

1 Introduction

Systems that integrate computing and communication capabilities to monitor and control physical processes are known as cyber-physical systems (CPSs) (Baheti and Gill; 2011). The increase in the use of communication networks for monitoring and control of physical systems also increases the vulnerability of CPSs to attacks in the network, which shows that the implementation of defense strategies is crucial for the reliable use of networked controlled systems.

Several works in the literature present strategies to detect and prevent the effects of cyber attacks considering different approaches (Mo and Sinopoli; 2010; Goes et al.; 2017; Sundaram and Hadjicostis; 2011; Pasqualetti et al.; 2013; Fawzi et al.; 2014). In the majority of these works, the system is modeled as a continuous-variable dynamic system.

Intrusion detection and prevention of damages caused by attacks in the context of supervisory control of discrete event systems are considered in Thorsley and Teneketzis (2006). The main objective of the work proposed in Thorsley and Teneketzis (2006) is to design a supervisor that achieves the specification in normal operation, and also after an attack. The attacks considered in Thorsley and Teneketzis (2006) can be interpreted as an interference in the communication channel between the supervisor and the plant. More recently, in Carvalho et al. (2016), the problem of intrusion detection and prevention in supervisory control systems, where the attacker has the ability to enable vulnerable actuator events that are disabled by the supervisor is addressed. A defense strategy that is capable of detecting attacks and disabling controllable events is proposed. In Carvalho et al. (2016), the defense strategy disables all controllable events of the system immediately after the detection of the attack.

In Lima et al. (2017), automaton models of the plant and supervisor under man-in-the-middle attacks in the sensor and/or communication channels are proposed. In addition, the property of NA-Safe controllability, that is related with the capability of detecting intrusions and preventing damages caused by man-in-the-middle attacks, is introduced, and a verification algorithm is proposed. However, in Lima et al. (2017), the authors do not present the implementation of a security module that detects and prevents the damages caused by cyber-attacks.

In this paper, we present a method for the implementation of a security module based on the definition of NA-Safe Controllability introduced in Lima et al. (2017). A small practical example is used to illustrate the implementation scheme, and to explain the basic ideas of the logic of protection

against cyber-attacks proposed in this work.

This paper is organized as follows. In Section 2, we present some preliminary concepts. In Section 3, we formulate the problem of intrusion detection and prevention of damages caused by man-in-the-middle attacks in CPSs. In Section 4, we define NA-Safe controllability. In Section 5, we present a necessary and sufficient condition for the existence of the security module, and a method for its online implementation. We also present in Section 5 a practical example, from modeling to implementation of the security module. Finally, in Section 6, the conclusions are drawn.

2 Preliminaries

Let $G = (X, \Sigma, f, x_0, X_m)$ be a deterministic automaton, where X is the state-space, Σ is the finite set of events, $f : X \times \Sigma \to X$ is the transition function, $x_0 \in X$ is the initial state of the system, and $X_m \subseteq X$ is the set of marked states. The set of marked states will be omitted unless stated otherwise. The domain of the transition function f can be extended to $X \times \Sigma^\star$, where $\Sigma^\star$ denotes the Kleene-closure of Σ, as follows: $f(x, \varepsilon) = x$, and $f(x, s\sigma) = f(f(x, s), \sigma)$, $\forall s \in \Sigma^\star$, and $\forall \sigma \in \Sigma$, where ε denotes the empty trace. The language generated by G is defined as $L(G) = \{s \in \Sigma^\star : f(x, s) \text{ is defined}\}$, and the set of active events of a state x of G is denoted as $\Gamma_G(x) = \{\sigma \in \Sigma : f(x, \sigma) \text{ is defined}\}$.

A nondeterministic automaton, on the other hand, is a five-tuple $\mathcal{G} = (X, \Sigma \cup \{\varepsilon\}, f_{nd}, x_0, X_m)$, where the elements of $\mathcal{G}$ have the same interpretation as in the deterministic automaton G, except for the nondeterministic transition function $f_{nd} : X \times \Sigma \cup \{\varepsilon\} \to 2^X$, and the set of initial states $x_0 \subseteq X$. In order to define the language generated by $\mathcal{G}$, it is necessary to extend the domain of f_{nd} to $X \times \Sigma^\star$, obtaining the extended transition function f^e_{nd}. Let $\varepsilon R(x)$ denote the ε-reach of a state x, *i.e.*, the set of states reached from x by following transitions labeled with ε, including state x (Cassandras and Lafortune; 2008). The ε-reach can be extended to a set of states $B \subseteq X$ as $\varepsilon R(B) = \cup_{x \in B} \varepsilon R(x)$. The extended nondeterministic transition function $f^e_{nd} : X \times \Sigma^\star \to 2^X$, can be defined recursively as $f^e_{nd}(x, \varepsilon) = \varepsilon R(x)$, and $f^e_{nd}(x, s\sigma) = \varepsilon R[\{z : z \in f_{nd}(y, \sigma) \text{ for some state } y \in f^e_{nd}(x, s)\}]$. Thus, the language generated by $\mathcal{G}$ can be defined as $L(\mathcal{G}) = \{s \in \Sigma^\star : (\exists x \in x_0)[f^e_{nd}(x, s) \text{ is defined}]\}$.

The prefix closure of a language $L \subseteq \Sigma^\star$ is defined as $\overline{L} = \{s \in \Sigma^\star : (\exists t \in \Sigma^\star)[st \in L]\}$. The post language after a trace s is defined as $L/s = \{t \in \Sigma^\star : st \in L\}$.

The set of events Σ can be partitioned as $\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$ where Σ_c and Σ_{uc} are, respectively, the sets of controllable and uncontrollable events of the system. The event set Σ can also be partitioned as $\Sigma = \Sigma_o \dot\cup \Sigma_{uo}$ where Σ_o and Σ_{uo} are the sets of observable and unobservable events of the system.

The projection $P_o : \Sigma^\star \to \Sigma_o^\star$, where $\Sigma_o \subset \Sigma$, is defined as $P_o(\varepsilon) = \varepsilon$, $P_o(\sigma) = \sigma$, if $\sigma \in \Sigma_o$ or $P_o(\sigma) = \varepsilon$, if $\sigma \in \Sigma \backslash \Sigma_o$, and $P_o(s\sigma) = P_o(s)P_o(\sigma)$, for all $s \in \Sigma^\star$ and $\sigma \in \Sigma$. The projection operation can be extended to languages by applying it to all traces in the language (Cassandras and Lafortune; 2008). The coaccessible part of G, denoted as $CoAc(G)$, is defined as usual (Cassandras and Lafortune; 2008).

Let G_1 and G_2 be two automata, then $G_1 \| G_2$ and $G_1 \times G_2$, denote, respectively, the parallel composition and the product of G_1 and G_2 (Cassandras and Lafortune; 2008).

Let $s \in L(G)$, then $|s|$ denotes the length of s. Let $\Sigma_s \subset \Sigma$, and let $s \in \Sigma^\star$. Then, with a slight abuse of notation, $\Sigma_s \in s$ denotes that at least one of the events that form s belongs to Σ_s.

3 Communication Network Attacks

In this paper we consider a networked supervisory control system, with attacks in communication channels and a Security Module, as shown in Figure 1. We assume that the communication between supervisor and plant is carried out by using several different channels. The channels that are used to send information, gathered by sensors, from the plant to the supervisor are denoted as sensor channels, and the channels that transmit the control actions, enabling actuators, from supervisor to plant are called supervisory control channels. In Figure 1, we can see the communication channels, where the physical bus and its connections with the devices are represented by solid lines, and the flow of information by dashed lines. The plant in the networked supervisory control system is modeled by a deterministic automaton $G = (X, \Sigma, f, x_0)$, the realization of the supervisor S is modeled by a deterministic automaton $H = (X_H, \Sigma, f_H, x_{0H})$, and the closed-loop system model $T = (X_T, \Sigma, f_T, x_{0T})$ is obtained by the parallel composition $T = G \| H$. We consider that the plant has a set of unsafe states denoted by $X_{US} \subset X$, and we assume that the supervisor is designed to avoid the plant from reaching any unsafe state $x \in X_{US}$, *i.e.*, no unsafe states are reachable in T.

Let us consider that there are vulnerable communication channels in the networked supervisory control system of Figure 1, and assume that the intruder can execute man-in-the-middle attacks, *i.e.*, the information in the attacked channel can be completely changed by the intruder. Let us also assume that both sensor communication channels and supervisory control communication channels, can be attacked. Thus, the intruder can hide, insert or replace events whose occurrence is de-

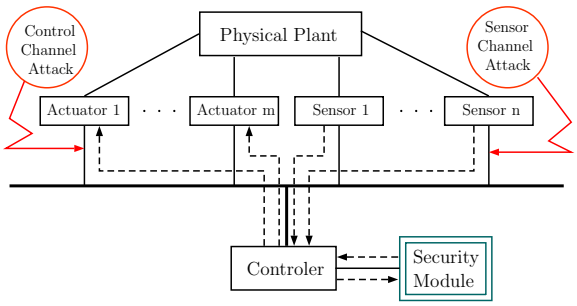

Figure 1: Closed-loop system under attack

tected by sensors in the plant, and can modify the enabling action commanded by the supervisor to the actuators of the plant connected through attacked channels, with the objective to drive the system to reach unsafe states. Let ch_{s_i} and ch_{a_j}, for $i = 1, \ldots, n_s$ and $j = 1, \ldots, n_a$, denote the attacked sensor channels and the attacked supervisory control channels, respectively, where n_s is the number of attacked sensor channels and n_a is the number of attacked supervisory control channels. Let $\Sigma_{s_i} \subset \Sigma_o$ and $\Sigma_{a_j} \subset \Sigma_c$, denote, respectively, the set of observable events transmitted through channel ch_{s_i} and the set of controllable events enabled through channel ch_{a_j}. Then, the set of events associated with the vulnerable sensor channels is defined as $\Sigma_{vs} = \sum_{i=1}^{n_s} \Sigma_{s_i}$, and the set of events associated with the vulnerable supervisory control channels is defined as $\Sigma_{va} = \sum_{j=1}^{n_a} \Sigma_{a_j}$.

In this paper, we consider that the supervisor has been designed and implemented in the system without considering possible network attacks. Thus, instead of changing all the control logic implemented in the system, we propose the design of a security module, as shown in Figure 1. These proposed Module observes the traces observed by the supervisor and, after detecting an attack that leads to unsafe states, prevent the system from reaching these states by forcing the supervisor to disable all controllable events of the system. Notice that, the communication between supervisor and Security Module it is not transmitted through the network, therefore it is not vulnerable to attacks. It is important to remark that Security Module allows that traces generated after network attacks, that do not belong to $T = G\|H$, be executed if these traces do not lead the system to reach unsafe states.

The following assumption is made in this work.

A1. *The sets of controllable and observable events are disjoint, i.e.,* $\Sigma_o \cap \Sigma_c = \emptyset$.

Assumption **A1** is necessary for the correct modeling of the plant and supervisor under network attacks. This assumption can always be relaxed by replacing the controllable and observable events $\sigma \in \Sigma_o \cap \Sigma_c$ with $\sigma_c\sigma_o$, where σ_c is controllable and unobservable, and σ_o is uncontrollable and observable, leading to new disjoint sets of controllable and observable events, as presented in Lima (2017).

4 NA-Safe Controllability

In order to prevent damages to the plant after network attacks, we propose in this paper the implementation of a module that is capable of identifying traces such that their continuations certainly lead the plant to reach an unsafe state,*i.e.*, a state in X_{US}, and then disable events to avoid the reaching of these unsafe states. In order to do so, a nondeterministic automaton that models the plant under attack $\mathcal{G}_a$, and a deterministic automaton that models the supervisor under attack H_a, must be computed as shown in Lima et al. (2017)[1]. Then, the model of the closed-loop system under attack is computed as $\mathcal{T}_a = \mathcal{G}_a\|H_a = (X_{\mathcal{T}_a}, \Sigma \cup \{\varepsilon\}, f_{\mathcal{T}_a}, x_{0,\mathcal{T}_a})$. Notice that, a state of $\mathcal{T}_a$ is composed of a state of $\mathcal{G}_a$, and a state of H_a. Thus, we can define the set of unsafe states of the attacked closed-loop system as $X_{US}^{\mathcal{T}_a} = \{(x, y) \in X_{\mathcal{T}_a} : x \in X_{US}\}$.

Since the malicious agent can enable attacked events in Σ_{va}, these events become uncontrollable. This leads to the following definition of the set of controllable events $\Sigma_{ca} = \Sigma_c \setminus \Sigma_{va}$, and of uncontrollable events $\Sigma_{uca} = \Sigma_{uc} \cup \Sigma_{va}$ of the attacked closed-loop system model $\mathcal{T}_a$.

Let $L(\mathcal{T}_a)$ denote the language generated by $\mathcal{T}_a$. Then, $L(\mathcal{T}_a)$ can be partitioned as $L(\mathcal{T}_a) = L_s(\mathcal{T}_a) \dot{\cup} L_{us}(\mathcal{T}_a)$, where $L_s(\mathcal{T}_a)$ denotes the safe language, composed of traces $s \in L(\mathcal{T}_a)$ such that it is not possible to reach an unsafe state in $X_{US}^{\mathcal{T}_a}$ after the occurrence of s, *i.e.*, $L_s(\mathcal{T}_a) = \{s \in L(\mathcal{T}_a) : (\forall t \in L(\mathcal{T}_a)/s)[f_{\mathcal{T}_a}^e(x_{0,\mathcal{T}_a}, st) \cap X_{US}^{\mathcal{T}_a} = \emptyset]\}$, and $L_{us}(\mathcal{T}_a)$ denotes the unsafe language, composed of traces $s \in L(\mathcal{T}_a)$ such that it is possible to reach an unsafe state in $X_{US}^{\mathcal{T}_a}$ after the occurrence of s, *i.e.*, $L_{us}(\mathcal{T}_a) = \{s \in L(\mathcal{T}_a) : (\exists t \in L(\mathcal{T}_a)/s)[f_{\mathcal{T}_a}^e(x_{0,\mathcal{T}_a}, st) \cap X_{US}^{\mathcal{T}_a} \neq \emptyset]\}$. Notice that a trace in the unsafe language may be part of the normal behavior of the system, *i.e.*, may belong to the language generated by T, $L(T)$, or even be part of $\overline{L}_s(\mathcal{T}_a)$. Therefore, the module may act only after distinguishing traces of $L_{\mathcal{T}_a}$ that will certainly reach an unsafe state, and do not belong to the language generated by the system before an attack $L(T)$. This leads to the following definition of NA-Safe Controllability.

Definition 1 (NA-Safe Controllability) *$L(\mathcal{T}_a)$ is said to be NA-Safe Controllable with respect to $P_o : \Sigma^\star \to \Sigma_o^\star$ and a set of unsafe states $X_{US}^{\mathcal{T}_a}$ if*

[1] In order to facilitate the readability of the paper the modeling technique proposed in Lima et al. (2017) is presented in Appendix A.

$(\forall s \in L(\mathcal{T}_a))[f^e_{\mathcal{T}_a}(x_{0,\mathcal{T}_a}, s) \cap X^{\mathcal{T}_a}_{US} \neq \emptyset] \Rightarrow (s = s_1 s_2)[(\forall \omega \in L(T) \cup \overline{L}_s(\mathcal{T}_a))[P_o(s_1) \neq P_o(\omega)] \wedge (\Sigma_{ca} \in s_2)]$. □

According to Definition 1, $L(\mathcal{T}_a)$ is NA-Safe Controllable if all traces $s \in L(\mathcal{T}_a)$, that lead the system to an unsafe state in $X^{\mathcal{T}_a}_{US}$, can be divided as $s = s_1 s_2$, such that: (i) $s_1 \in L_{us}(\mathcal{T}_a)$ can be distinguished from any trace of $L(T)$ and $\overline{L}_s(\mathcal{T}_a)$; and (ii) $s_2 \in \Sigma^\star$ has an event from Σ_{ca}. These two conditions allow that unsafe traces that certainly lead the system to an unsafe state be detected, and the reach of unsafe states prevented by disabling the controllable events of Σ_{ca}.

Notice that, the supervisor could be designed to disable all controllable events of the plant after detecting the observation of traces that do not belong to $P_o(L(T))$. However, this approach would be more restrictive than the approach proposed in this paper, since the system can execute safe traces in $\overline{L}_s(\mathcal{T}_a)$ after an attack that do not represent danger to the system.

An algorithm for the verification of NA-Safe Controllability of the language of the attacked system $L(\mathcal{T}_a)$ is presented in Lima et al. (2017).

5 Implementation of the Security Module

We state in the sequel that the NA-Safe Controllability of $L(\mathcal{T}_a)$ is a necessary and sufficient condition for the existence of the security module (Lima et al.; 2017).

Theorem 1 *There exists a security module that is capable of preventing G from reaching an unsafe state in X_{US} if, and only if, $L(\mathcal{T}_a)$ is NA-Safe Controllable with respect to $P_o : \Sigma^\star \to \Sigma_o^\star$ and $X^{\mathcal{T}_a}_{US}$.*

According to Theorem 1, if $L(\mathcal{T}_a)$ is NA-Safe Controllable, then the security module implemented would always prevent damage in the system. Notice that the security module observes the same trace observed by the supervisor, and then it verifies if this trace belongs to $P_o(L(T) \cup \overline{L}_s(\mathcal{T}_a))$. If the observed trace does not belong to $P_o(L(T) \cup \overline{L}_s(\mathcal{T}_a))$, then the security module must send an information to the supervisor to disable all controllable events.

In order to construct the security module, we need first to compute automaton $\mathcal{T}_S$ whose generated language is $L(T) \cup \overline{L}_s(\mathcal{T}_a)$. This automaton is construct as shown in Algorithm 1.

Algorithm 1 *Construction of $\mathcal{T}_S$.*

Inputs: $T = (X_T, \Sigma, f_T, x_{0,T})$, $\mathcal{T}_a = (X_{\mathcal{T}_a}, \Sigma \cup \{\varepsilon\}, f_{\mathcal{T}_a}, x_{0,\mathcal{T}_a})$, Σ_{ca}, $X^{\mathcal{T}_a}_{US}$.
Output: $\mathcal{T}_S = (X_{\mathcal{T}_S}, \Sigma \cup \{\varepsilon\}, f_{\mathcal{T}_S}, x_{0,\mathcal{T}_S})$

1: Define $X^{\mathcal{T}_a}_{US}$ as the set of marked states of $\mathcal{T}_a$.

2: Define $\mathcal{T}_U = CoAc(\mathcal{T}_a) = (X_U, \Sigma \cup \{\varepsilon\}, f_U, x_{0,U}, X^{\mathcal{T}_a}_{US})$, and unmark its states.

3: Define $X_{\mathcal{T}_a} \setminus X_U$ as the set of marked states of $\mathcal{T}_a$.

4: Define $\mathcal{T}'_S = CoAc(\mathcal{T}_a)$, and unmark its states.

5: Construct automaton $\mathcal{T}_S$ such that $L(\mathcal{T}_S) = L(\mathcal{T}'_S) \cup L(T)$.

After the computation of automaton $\mathcal{T}_S$, the security module operation can be implemented as shown in Algorithm 2.

Algorithm 2 *Security module operation*

1: Compute the initial state estimate of automaton $\mathcal{T}_S$, $x_{0,Obs}$, and define the current state estimate as $x_{c,Obs} = x_{0,Obs}$.

2: Define the set of events Γ_c in the current state estimate $x_{c,Obs}$ that belong to the active events $\bigcup_{x \in x_{c,Obs}} \Gamma(x)$.

3: Wait for the next event observation e:

3.1: If $e \notin \Gamma_c$, disable all controllable events of Σ_{ca}.

3.2: Else, update the current state estimate $x_{c,Obs}$ of automaton $\mathcal{T}_S$, and go back to Step 2.

In order to illustrate the implementation of the security module following the method described in Algorithm 2, let us consider an example of a railway system shown in Figure 2, which consists of two tracks connected with a secondary line where trains can move in these tracks. The secondary line connect both tracks and allow Train 2 to move to Track 1, once Train 2 is in the secondary line it can also return to Track 2. In Figure 2 sensors are represented by arrows, and identify the passing of a train through a position in the track. The switches are represented by red lines in tracks, and change the direction of trains.

The railway system desired behavior is that Train 1 moves only in Track 1, and Train 2 moves only in Track 2. Thus, the supervisor objective is to avoid the crashing of the trains, *i.e.*, the objective is to keep Train 2 in Track 2.

In this example, the set of observable events is given by $\Sigma_o = \{a, b, d, e\}$, that are associated with observation by sensors placed in the tracks. The set of controllable events is given by $\Sigma_c = \{a, g, c\}$, where a and g are associated with switches such that when a is enabled, Train 2 is deviated to the secondary line, and when g is enabled, Train 2

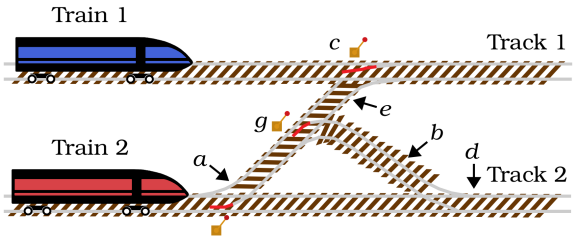

Figure 2: Railway example.

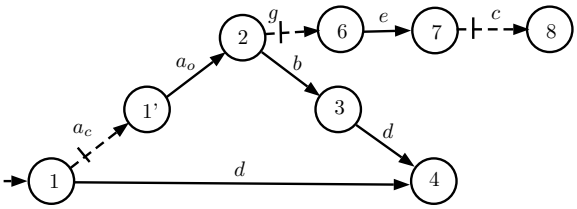

Figure 4: Similar plant model $\tilde{G}$ obtained from G for the railway example.

moves in the direction of Track 1. Event c represents the entrance of Train 2 in Track 1. Thus, when event c is disabled, Train 2 cannot enter in Track 1. This can be done, for instance, by turning off Train 2 or derailing it. Notice, therefore, that event c must be enabled during the non attacked operation of the system.

With a view to designing the security module for the system of Figure 2, it is first necessary to obtain the automaton model for the plant, depicted in Figure 3. Each state of G represents a position of Train 2 on the railway, such that state 8 represents that Train 2 has reached Track 1 and, therefore, is an unsafe state, $X_{US} = \{8\}$.

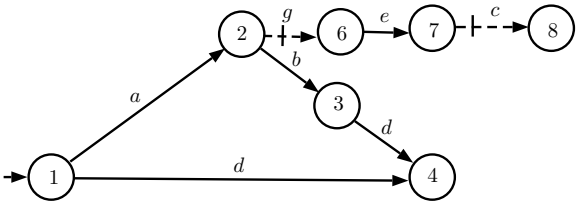

Figure 3: Automaton model G.

In order to implement a security module, assumption **A1** must be valid. However, in this case, this assumption is violated, since event a is controllable and observable. This problem can be easily circumvented by replacing event a with trace $a_c a_o$, where a_c is controllable and unobservable and a_o is observable and uncontrollable, and by constructing a similar system $\tilde{G}$, represented in Figure 4, whose set of controllable events is given by $\tilde{\Sigma}_c = \{a_c, c, g\}$ and the set of observable events is given by $\tilde{\Sigma}_o = \{a_o, b, d, e\}$. As shown in Lima (2017), $\tilde{G}$ generates the same observed language as G, and satisfies Assumption **A1**.

A supervisor H that keeps both trains in their

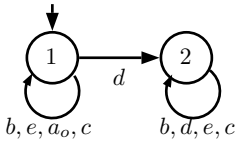

Figure 5: Supervisor H.

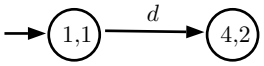

Figure 6: Closed-loop system T for the railway example.

respective tracks, avoiding Train 2 to reach Track 1, is shown in Figure 5. Thus, the closed-loop system $T = \tilde{G} \parallel H$ is presented in Figure 6 where Train 2 stays in Track 2. Notice that, as expected, the unsafe state 8 does not belong to the non attacked system behavior.

Let us consider now that the set of vulnerable observable events is given as $\Sigma_{vs} = \{a_o\}$. Thus, following Algorithm 3 presented in Appendix A, that was proposed in Lima et al. (2017). The attacked plant model $\mathcal{G}_a$ is represented in Figure 7. Notice that self loops labeled with the vulnerable events are added to each state of $\tilde{G}$ to represent the capability of the malicious agent of creating observations of an vulnerable event. Moreover, transitions labeled with ε are added in parallel to the transitions labeled with vulnerable events in order to represent the capability of the intruder of hiding events. The capability of changing an observation has the same effect as hiding an event and then creating a new event. Thus, this type of modification is already represented by the self loop and ε transitions added to $\tilde{G}$.

Let $\Sigma_{va} = \{a_c, g\}$ be the set of vulnerable actuator events. Then, the supervisor under attack H_a is represented in Figure 8. For the construction of the attacked supervisor we used the Algorithm 4 presented in Appendix A that was proposed in Lima et al. (2017), where the supervisory channel attack is modeled by adding a self loop labeled with the vulnerable actuator events in the states of H to represent that these events are uncontrollable after the attack.

The resulting attacked closed-loop system $\mathcal{T}_a = \mathcal{G}_a \| H_a$ is represented in Figure 9. No-

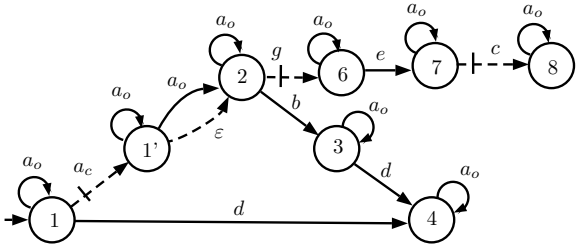

Figure 7: Plant model under attack $\mathcal{G}_a$.

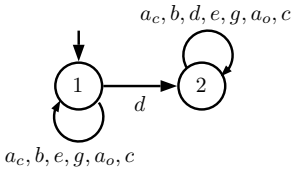

Figure 8: Supervisor model under attack H_a.

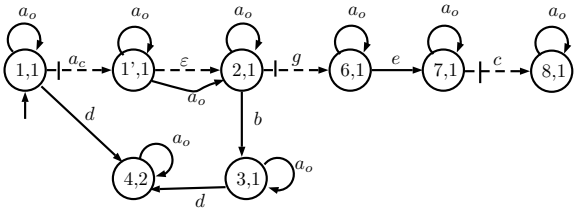

Figure 9: Closed-loop system model under attack $\mathcal{T}_a$ for the railway example.

tice that state $(8,1)$ represents that Train 2 has reached Track 1, *i.e.*, the attack can make the system reach the unsafe state 8. In order to verify if it is possible to avoid reaching the unsafe state by implementing the security module, the verification algorithm proposed in Lima et al. (2017) can be used. In this example, the language generated by $\mathcal{T}_a$, $L(\mathcal{T}_a)$, is NA-Safe Controllable, which according to Theorem 1, implies that it is possible to implement the security module that avoids the plant from reaching state 8.

In order to implement the security module, it is first necessary to compute automaton $\mathcal{T}_S$ following the steps of Algorithm 1. In Steps 1 and 2 of Algorithm 1, we compute automaton $\mathcal{T}_U$, depicted in Figure 10, by removing from $\mathcal{T}_a$ the states from which it is not possible to reach the unsafe state $(8,1)$, *i.e.*, in this example, states $(4,2)$ and $(3,1)$. The computation of automaton $\mathcal{T}_S'$, depicted in Figure 11, is carried out by following Steps 3 and 4 of Algorithm 1. In this example, states $(6,1)$, $(7,1)$, and $(8,1)$ are removed from $\mathcal{T}_a$ to obtain automaton $\mathcal{T}_S'$. Now, following Step 5 of Algorithm 1, $\mathcal{T}_S = \mathcal{T}_S'$ is constructed.

As shown in Algorithm 2, the security module performs the online state estimate of $\mathcal{T}_S$ after the observation of the events executed by the plant. If the observed event is not feasible in the current state estimate, then the attack is detected and all non vulnerable controllable events are disabled by the security module.

In order to illustrate the security scheme proposed in this paper, let the observer of $\mathcal{T}_S$ be

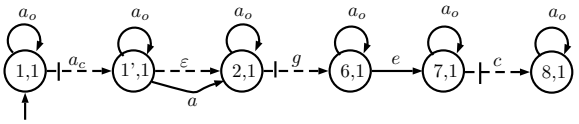

Figure 10: Unsafe automaton model $\mathcal{T}_U$.

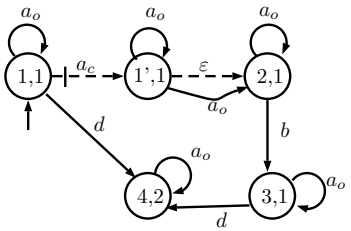

Figure 11: Safe automaton model $\mathcal{T}_S = \mathcal{T}_S'$.

presented in Figure 12. Now, consider that the system is in its initial state 1, and the current state estimate of $\mathcal{T}_S$ is in the initial state $\{(1,1),(1',1),(2,1)\}$. Then, let us consider that the intruder enables event a_c, which makes Train 2 deviates to the secondary line. Then, the observation of event a_o is erased by the intruder, and the security module is not capable of detecting the intrusion. Notice that in state $\{(1,1),(1',1),(2,1)\}$ the feasible events of the observer of $\mathcal{T}_S$ are a_o, b, and d. If the attacker does not enable event g, then Train 2 goes back to Track 2, and event b is observed. In this case, the security module does not force the disablement of the non vulnerable controllable events in Σ_{ca}. However, if event g is enabled by the intruder after trace $a_c a_o$ has been executed by the plant, then event e is observed by the security module when Train 2 crosses the switch associated with g. Since e is not feasible in the current state estimate of $\mathcal{T}_S$, then event c is disabled by the security module, and Train 2 is stopped without reaching Track 1. This shows that the plant will never reach state 8 due to the action of the security module.

Notice that if disabling event c is also not desired, because it may represent, for instance, the derail of Train 2 by the security module, then, the supervisory control communication channel associated with event g must have a higher level of protection against attacks. The verification of which communication channels must have a higher level of hardware protection against cyber attacks can also be addressed using the method proposed in this paper.

It is important to remark that in this small example, it is not difficult to obtain the protection logic against cyber attacks without using the strategy proposed in this paper. However, this can be a very difficult task for large and complex systems. In these cases, the method proposed in this paper for the design and implementation of a security module can be used.

6 Conclusions

In this paper, we propose the implementation of a security module that is capable of preventing damages caused by man-in-the-middle attacks in sensor and/or control communication channels in

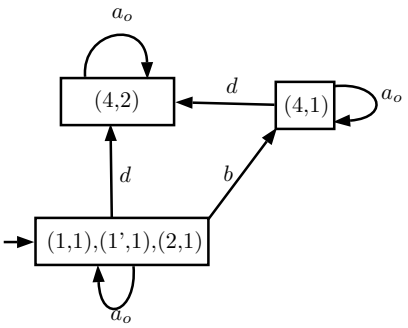

Figure 12: Observer of $\mathcal{T}_S$.

Cyber-Physical Systems. A small practical example is used to illustrate the implementation of the security module, and to explain the protection logic.

References

Baheti, R. and Gill, H. (2011). Cyber-physical systems, *in* T. Samad and A. Annaswamy (eds), *The Impact of Control Technology*, IEEE Control Systems Society.

Carvalho, L. K., Wu, Y.-C., Kwong, R. and Lafortune, S. (2016). Detection and prevention of actuator enablement attacks in supervisory control systems, *2016 13th International Workshop on Discrete Event Systems (WODES)*, Xi'an, China, pp. 298–305.

Cassandras, C. G. and Lafortune, S. (2008). *Introduction to discrete event systems*, Springer.

Fawzi, H., Tabuada, P. and Diggavi, S. (2014). Secure estimation and control for cyber-physical systems under adversarial attacks, *IEEE Transactions on Automatic Control* **59**(6): 1454–1467.

Goes, R. M., Kang, E., Kwong, R. H. and Lafortune, S. (2017). Stealthy deception attacks for cyber-physical systems, *Proceedings of the 56th IEEE Conference on Decision and Control*, Melbourne, Australia, pp. 4224–4230.

Lima, P. M. (2017). *Security against network attacks in supervisory control systems*, Master's thesis, Federal University of Rio de Janeiro.

Lima, P. M., Alves, M. V., Carvalho, L. K. and Moreira, M. V. (2017). Security against network attacks in supervisory control systems, *Proceedings of the 20th IFAC World Congress*, Vol. 50, Toulouse, France, pp. 12333–12338.

Mo, Y. and Sinopoli, B. (2010). False data injection attacks in control systems, *Preprints of the 1st workshop on Secure Control Systems*, Stockholm, Sweden, pp. 1–6.

Pasqualetti, F., Dörfler, F. and Bullo, F. (2013). Attack detection and identification in cyber-physical systems, *IEEE Transactions on Automatic Control* **58**(11): 2715–2729.

Sundaram, S. and Hadjicostis, C. N. (2011). Distributed function calculation via linear iterative strategies in the presence of malicious agents, *IEEE Transactions on Automatic Control* **56**(7): 1495–1508.

Thorsley, D. and Teneketzis, D. (2006). Intrusion detection in controlled discrete event systems, *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, USA, pp. 6047–6054.

A Modeling of the attacked plant and supervisor

The modeling of the plant and supervisor under cyber-attacks is presented in Lima et al. (2017). In order to facilitate the readability of this paper, we present in the sequel Algorithms 3 and 4, that compute automata $\mathcal{G}_a$ and H_a, respectively.

Algorithm 3 *Computation of automaton $\mathcal{G}_A$ that models the plant subject to sensor channel attacks*

Input: $G = (X, \Sigma, f, x_0)$, $\Sigma_{vs} \subseteq \Sigma_o$.
Output: $\mathcal{G}_a = (X, \Sigma \cup \{\varepsilon\}, f_a, x_0)$.

1: Define $f_a(x, \sigma)$, $\forall x \in X$, and $\forall \sigma \in \Sigma \cup \{\varepsilon\}$, as follows:

$$f_a(x,\sigma) = \begin{cases} \{f(x,\sigma)\}, \text{ if } \sigma \in \Gamma_G(x) \cap (\Sigma \setminus \Sigma_{vs}), \\ \{x\} \cup \{f(x,\sigma_v) : \sigma_v \in \Gamma_G(x) \cap \Sigma_{vs}\}, \text{ if } \sigma \in \Sigma_{vs}, \\ \{f(x,\sigma_v) : \sigma_v \in \Gamma_G(x) \cap \Sigma_{vs}\}, \text{ if } \sigma = \varepsilon, \\ \text{undefined, otherwise.} \end{cases}$$

Algorithm 4 *Computation of automaton H_a that models the supervisor subject to control channel attacks*

Input: $H = (X_H, \Sigma, f_H, x_{0H})$, $\Sigma_{va} \subseteq \Sigma_c$.
Output: $H_a = (X_H, \Sigma, f_{H_a}, x_{0H})$.

1: Define $f_{H_a}(x, \sigma)$, $\forall x \in X_H$, and $\forall \sigma \in \Sigma$ as follows:

$$f_{HA}(x,\sigma) = \begin{cases} f_H(x,\sigma), \text{ if } \sigma \in \Gamma_H(x) \\ x, \text{ if } \sigma \in (\Sigma_{uc} \cup \Sigma_{va}) \setminus \Gamma_H(x) \\ \text{undefined, otherwise.} \end{cases}$$

Thanks

This work has been partially supported by CNPq and CAPES.