Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 53-4 (2020) 7–13

Detection and Prevention of Cyber-Attacks in Networked Control Systems [⋆]

Yike Li [*] **Yin Tong** [*] **Alessandro Giua** [**]

[*] *SIST, Southwest Jiaotong University, Chengdu 611756, China (e-mail: yikeli@my.swjtu.edu.cn; yintong@swjtu.edu.cn).*
[**] *DIEE, University of Cagliari, Cagliari 09123, Italy (e-mail: giua@unica.it)*

Abstract: This paper addresses the problem of detection and prevention of cyber attacks in discrete event systems where the supervisor communicates with the plant via network channels. Random control delays may occur in such networked systems, hence the control of the supervisor could be affected. Furthermore, there is an attacker targeting the vulnerable actuators. The attacker can corrupt the control input generated by the supervisor, and aims at driving the plant to unsafe states. We propose a new approach to model the closed-loop system subject to control delays and attacks. The notion of AE-safe controllability in the networked control system is defined: it describes the ability to prevent the plant from reaching unsafe states after attacks are detected. A method for testing AE-safe controllability is also presented.

Keywords: Discrete event system; Cyber security; Supervisory control; Networked control system.

1. INTRODUCTION

A cyber-physical system (CPS) is a mechanism controlled or monitored by means of computer-based algorithms, often tightly integrated with a public network and its users. Applications of CPS have higher risks of suffering from specific vulnerabilities that do not exist in classical control systems (Pasqualetti et al., 2013). A cyber-attack is one of those critical threats, which has become increasingly sophisticated and dangerous.

In discrete event systems (DESs), supervisors are usually synthesized to guarantee opacity, safety, etc. (Tong et al., 2018; Yin and Lafortune, 2015). However, controlled systems may be also vulnerable to malicious attacks and this should be taken into account in the design and verification phases. In Lima et al. (2017), the so-called man-in-the-middle attack is considered. The authors also propose a defense strategy to detect such attacks and to prevent the damages caused by the attacks. Su (2017) and Zhang et al. (2019) consider attackers that can “fool” the supervisor by altering the observation and make the supervisor lead the system to unsafe states. Carvalho et al. (2018) address the detection and prevention of four classes of attacks: Actuator Enablement attacks (AE-attacks), Actuator Disablement attacks (AD-attacks), Sensor Erasure attacks (SE-attacks) and Sensor Insertion attacks (SI-attacks). In Carvalho et al. (2016) a particular case of AE-attacks is considered where the attacker changes the control input by enabling events that are disabled by the supervisor, and the property of *AE-safe controllability* is defined. It characterizes the system’s capability to detect attacks and then to prevent the system from reaching unsafe states through disabling some controllable events.

The networked structure is a significant feature of CPS. In networked control systems (NCSs), the plant to be controlled communicates with the supervisor through public networks. Nevertheless, such a networked structure may be affected by communication delays and losses as the result of distance, unstable hardware and other factors. Therefore, the networked system paradigm captures phenomena that are common in practical implementations. Lin (2014) has developed a formal approach for supervisory control in networked discrete event systems. The networked supervisor guarantees that the closed-loop system still satisfies the specification even when observations and control inputs are delayed or lost. However, to the best of our knowledge, the problems of detection and prevention of cyber-attacks in NCSs have never been explored.

Note that the networked control problem with observation delays and losses can be reduced to the standard supervisory control problem by constructing the delayed observer. Herein we focus on the detection and prevention of cyber-attacks in NCSs and only the control delays are explicitly taken into account. In particular, we consider a closed-loop system as depicted in Figure. 1. The plant is modeled as a deterministic finite automaton (DFA), which is observed and controlled by the supervisor via the network. There are two communication channels: the observation channel and the control channel, between the plant and the supervisor. In the observation channel there is no communication delay but in the control channel the communication delay is bounded by M, a non-negative integer. Some actuators are subject to AE-attacks, i.e., the attacker can change supervisor’s control input on some

[⋆] (Corresponding Author: Y. Tong) This work was supported by the National Natural Science Foundation of China under Grant No. 61803317, and the Fundamental Research Funds for the Central Universities under Grant No. 2682018CX24.

2405-8963 Copyright © 2020 The Authors. This is an open access article under the CC BY-NC-ND license.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2021.04.001

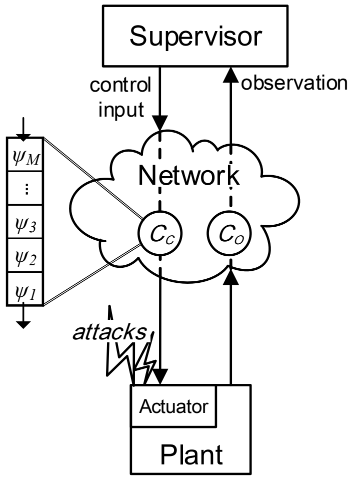


Fig. 1. The architecture of the NCS under attacks

vulnerable actuators from "disable" to "enable". We aim at determining after the attacks are detected whether it is possible to disable some controllable events so that the system is protected from reaching unsafe states.

The main challenge in NCSs with control delays is that the standard automaton realization of a supervisor does not model the possible delays of control inputs, which makes concurrent composition unfeasible to construct the model of the closed-loop system. In Lin (2014), only a recursive way to compute the generated language of the closed-loop system is presented. However, when the problem of detection and prevention of cyber-attacks is considered, it is necessary to construct the model of the closed-loop system since attacks are detected based on the estimation of all possible behavior of the system. The main contributions of the paper are: i) AE-safe controllability in NCSs is defined; ii) An algorithm for constructing the closed-loop system considering control delays is proposed; iii) An algorithm for constructing the AE-attacked model is presented; iv) Finally, an algorithm for verifying AE-safe controllability in NCSs is developed.

2. PRELIMINARIES

2.1 Automaton model

A *deterministic finite automaton* (DFA) is a four-tuple $G = (X, E, \delta, x_0)$, where X is a set of states, E is a set of events (alphabet), $\delta : X \times E \to X$ is a (possibly partial) transition function, and $x_0 \in X$ is an initial state. The transition function is also usually extended to $\delta : X \times E^* \to X$. In *nondeterministic finite automata* (NFA), $\delta(x, e)$ may be a subset of states (as opposed to a single one), i.e., $\delta(x, e) \subseteq X$, and $e \in E \cup \{\varepsilon\}$. The *generated language* of G is $L(G) = \{s \in E^* : \delta(x_0, s) \text{ is defined}\}$. Given a language $L \subseteq E^*$ and a string $s \in L$, we define $L/s = \{s' \in E^* | ss' \in L\}$ the set of *suffix* strings of s. If there exists $s'' \in L/s'$ such $s's'' = s$, s' is a *prefix* of s, denoted as $s' \preceq s$. In addition, if $s' \neq s$, then s' is a *strict prefix* of s, denoted as $s' \precneqq s$. If a string $s \in E^*$ contains an event in $E' \subseteq E$, we denote $E' \vdash s$; Otherwise, we denote $E' \nvdash s$. An event $e \in E$ is said to be *active* at state $x \in X$, if $\delta(x, e)$ is defined. The set of events active at state x is denoted as $\mathcal{A}(x)$.

The event set E can be partitioned as $E = E_o \dot{\cup} E_{uo}$, where E_o and E_{uo} are the sets of observable and unobservable events, respectively. The natural projection $P_o : E^* \to E_o^*$ is defined as (i) $P_o(\varepsilon) = \varepsilon$; (ii) $P_o(e) = e$, if $e \in E_o$; (iii) $P_o(e) = \varepsilon$, if $e \in E_{uo}$; (iv) $P_o(se) = P(s)P(e)$, $s \in E^*, e \in E$.

2.2 Networked supervisory control

Given a plant $G = (X, E, \delta, x_0)$, the goal of supervisory control is to design a control agent (called supervisor) that restricts the behavior of the plant within a set of safe (or legal) states. The supervisor observes a set E_o of observable events and is able to control a set $E_c \subseteq E$ of controllable events. Therefore, the set E of events may also be partitioned as $E = E_c \dot{\cup} E_{uc}$, where E_c and E_{uc} denote the sets of controllable and uncontrollable events respectively. In this section, we introduce the framework of supervisory control for networked systems presented by Lin (2014). In the networked setting, the supervisor communicates with the plant via two channels, that is, the observation channel C_o and the control channel C_c. Here it is assumed that: (1) Message transmitted in C_o and C_c satisfy FIFO policy. (2) In C_o there is no communication delay. (3) Control inputs transmission in C_c may be affected by a random delay which is bounded by $M \in \{0, 1, 2, \ldots\}$ (finite) steps. (4) At the initial state, the control input is applied immediately without delays.

Assumptions (1), (3) and (4) are common in the study of networked discrete event systems. We add Assumption (2) in order to focus on the issues caused by control delays. We believe that if observation delays are considered, the proposed approaches can be applied with proper modifications. Network observability and network controllability were defined in Lin (2014), where it was shown that there exists a *state-estimate-based networked supervisor* such that the NCS satisfies the specification if and only if the specification is network controllable and network observable. Since the synthesis of the networked supervisor is not our focus, here we only briefly explain how it works.

The state-estimate-based networked supervisor is formalized as a mapping $\psi : 2^X \to 2^E$: given an observation w and its estimate $\mathcal{X}(w)$, the supervisor will issue a control input $\psi(\mathcal{X}(w)) \supseteq E_{uc}$ that contains events enabled by the supervisor. However, due to delays in the control channel, the control input may not be received by the actuator immediately. In this case, the actuator will use the control input $\psi(\mathcal{X}(w'))$ issued $i \in \{0, 1, \ldots, N\}$ steps before, where $w' \preceq w$ is obtained by removing the last i events in w. Even so, the supervisor guarantees that the controlled system will never reach the given set $X_f \subsetneq X$ of forbidden (or unsafe) states provided that the specification is networked controllable and observable. Since the estimate $\mathcal{X}(w)$ is a function of the observation, the supervisor can be reduced to a function $\psi : E_o^* \to 2^E$ and be represented as a DFA $H = (Z, E, \delta_h, z_0)$. Given an observation w, the corresponding control input $\psi(w)$ is "encoded" into a state in H. Namely, $\psi(w) = \mathcal{A}(z) \cup E_{uc}$, where $z = \delta_h(z_0, w)$. Therefore, we refer $\mathcal{A}(z)$ as the control input implying that all uncontrollable events are also included. We use the following example to further illustrate the dynamics of G under the control of H when control delays are considered.

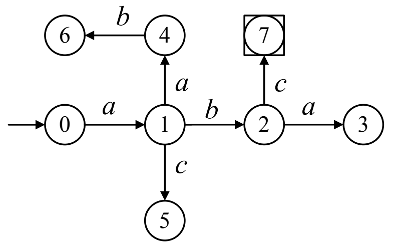
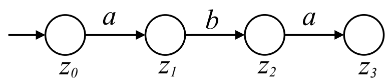
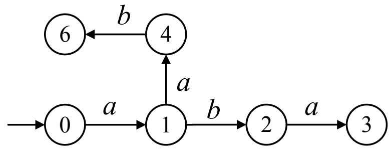


Fig. 2. The plant G, where $E_o = \{a, b\}$ and $E_c = \{a, c\}$.



(a) H



(b) H/G

Fig. 3. The automaton representation of ψ (a) and the model of the network controlled system (b).

Example 1. Let us consider the plant G in Fig. 2. Let $E_o = \{a, b\}$, $E_c = \{a, c\}$, and $X_f = \{7\}$. Let $M = 2$ be the upper bound of control delays. The automaton in Fig. 3(a) represents a networked supervisor ψ such that state 7 is never reachable in the closed-loop system.

Initially, nothing is observed. From H, the control input for $w = \varepsilon$ is $\mathcal{A}(z_0) = \{a, b\}$ and it is received by the plant immediately. Therefore, with event a occurring next, the plant reaches state 1 from state 0, and the supervisor accepts a and moves to state z_1 where a new control input $\mathcal{A}(z_1) = \{b\}$ is issued. There are two possibilities:

- If the control input $\mathcal{A}(z_1)$ is delayed, the plant will adopt $\mathcal{A}(z_0)$. Then, if a occurs, reaches state 4. Note that H does not accept a at state z_1 and no control input is issued. The control input in use is still $\mathcal{A}(z_0)$ where b is enabled. With the occurrence of b, the plant reaches state 6 from 4.
- If the control input $\mathcal{A}(z_1)$ is not delayed (or, even if delayed) and it reaches the plant before the occurrence of the next a, the plant adopts $\mathcal{A}(z_1)$ and eventually, on the occurrence of b, reaches state 2 from state 1.

Considering all possible delays and repeating the procedure, we obtain the closed-loop system in Fig. 3(b). Clearly, it is different from the concurrent composition of H and G. We also point out that disabling event c at state z_1 guarantees that even if the control input $\mathcal{A}(z_2) = \{a, b\}$ is delayed while the plant is at state 2 and using $\mathcal{A}(z_1)$ or $\mathcal{A}(z_0)$, the plant will not reach state 7 through event c. $\diamond$

Example 1 shows that the DFA H is a finite representation of function ψ but does not model the control delays. Therefore, the model of the controlled system H/G cannot be constructed via the standard concurrent composition between H and G.

3. MODEL OF THE CLOSED-LOOP SYSTEM

In this section, we propose an algorithm to construct the automaton model of H/G.

Since the model H of the supervisor does not contain control delays, we first propose an algorithm to construct a new automaton realization H_M in which the control delays are described as well. Given a supervisor $H = (Z, E, \delta_h, z_0)$ and the upper bound M on the control delay, $H_M = (Q, E \cup \{\varepsilon\}, \delta_m, q_0)$ is an NFA, where $Q \subseteq (Z \cup \{\text{-}\})^{M+1} \times \{1, 2, \ldots, M+1\}$ and $q_0 = (\text{-}, \text{-}, \ldots, \text{-}, z_0|M+1)$. A state q of H_M is a $M+2$ tuple. Each of its first $M+1$ elements is either a state in H or the symbol "-", which means null. The first $M+1$ elements of q represent the last $M+1$ control inputs generated by the supervisor, which are the possible control input that can be used by the plant. The last element is an integer from 1 to $M+1$ specifying which control input generated by the supervisor, among the last $M+1$, has currently been received by the plant. The occurrence of an event $e \in E$ in H_M means the event is received by the supervisor while the most recent new control input $\mathcal{A}(z)$ has not been received by the plant. The empty word ε in H_M represents that the plant receives a new control input from the FIFO queue, thus effectively reducing the current delay.

We denote $q(i)$ the i-th element of q for $i \in \{1, \ldots, M+2\}$ and add a "|" to graphically separate the first $M+1$ elements and the last one. Let $\delta_m(q_0, w) = q = (q(1), q(2), \ldots, q(M), q(M+1)|k)$ be a state in H_M reached through $w \in E^*$.

- For $i \in \{1, \ldots, M\}$, $q(i) \in Z \cup \{\text{-}\}$ is the state reached $M+1-i$ steps before $q(M+1)$, i.e., $q(i) = \delta_h(z_0, w_i)$, where $w_i \preceq w$ with the first i events kept. If $M + 1 - i > |w|$ (i.e., w is shorter than M), $q(i) =$ -. The queue $q(1), q(2), \ldots, q(M)$ represents the queue of states corresponding to the control inputs that may be delayed.
- $q(M+1) \in Z$ is the current state of H. Namely, $q(M+1) = \delta_h(z_0, w)$.
- $k \in \{1, 2, \ldots, M+1\}$ describes the position of the state whose corresponding control input $\mathcal{A}(q(k))$ is currently received by the plant among $q(1), q(2), \ldots,$ $q(M), q(M+1)$. Namely, the control input in use was issued $M+1-k$ steps before.

Clearly, if $k = M+1$, i.e., there is no control input is delayed, event ε is not active at q; If $k = 1$, i.e., the most recent new control input cannot be delayed anymore, no event in E is active at q; Otherwise, events in E and ε may be both active at q. Therefore, given a state $q = (q(1), q(2), \ldots, q(M), q(M+1)|k)$ and $e \in E \cup \{\varepsilon\}$, the transition function $\delta_m(q, e)$ are determined as follows.

- If $k = M+1$: Only $\delta_m(q, e) := q'$ is defined for all $e \in \mathcal{A}(q(M+1))$, where $q'(i) := q(i+1)$ for $i = 1, 2, \ldots, M$, $q'(M+1) := \delta_h(q(M+1), e)$ and $q'(M+2) := k-1$. In words, all the elements $q(2), \ldots, q(M+1)$ move one position forward in q', the current state of H is updated to $\delta_h(q(M+1), e)$ and the control input is delayed one step as $q'(M+2) = k-1$.

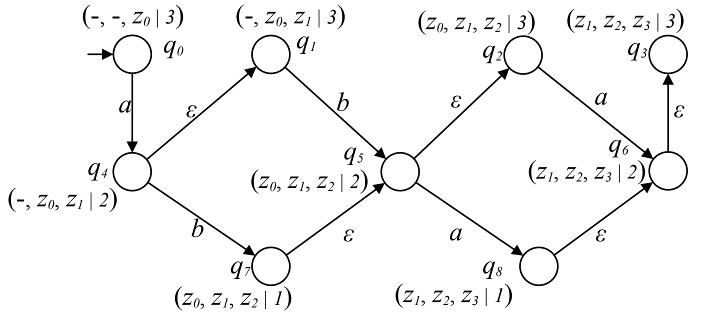


Fig. 4. The automaton model H_M of the networked supervisor.

- If $k = 1$: Only $\delta_m(q, \varepsilon) := q''$ is defined, where $q''(i) := q(i)$ for $i = 1, 2, \dots, M+1$, and $q''(M+2) := k+1$.
- Otherwise: Both $\delta_m(q, e) := q'$ and $\delta_m(q, \varepsilon) := q''$ are defined, where $e \in \mathcal{A}(q(M+1))$, q' and q'' are computed following the same definitions above.

Example 2. Consider the NCS in Example 1. Fig. 4 shows the automaton model H_M of the networked supervisor constructed. Initially, $q_0 = (\text{-}, \text{-}, z_0|3)$. Event a is defined at z_0, therefore one transitions is added: $\delta_m(q_0, a) = q_4 = (\text{-}, z_0, z_1|2)$. For q_4, both $\delta_m(q_4, b) = q_7$ and $\delta_m(q_4, \varepsilon) = q_1$ are defined. ◇

Based on H_M, the closed-loop system H/G can be constructed through a synchronization operation $\otimes$ between G and H_M. The obtained automaton is an NFA, denoted as $G_R = G \otimes H_M = (R, E, \delta_r, r_0)$, where $R \subseteq X \times Q$, $r_0 = (x_0, q_0)$ and the transition relation $\delta_r(r, e)$ for $r = (x, q) \in R$ and $e \in E \cup \{\varepsilon\}$ is defined as follows:

$$\delta_r(r,e) = \begin{cases} (\delta(x,e), \delta_m(q,e)), & \text{if } e \in \mathcal{A}(x) \cap \mathcal{A}(q) \cap \\ & \mathcal{A}(q(q(M+2))); \\ (\delta(x,e), q), & \text{if } e \in \mathcal{A}(q(q(M+2))) \cap \\ & (\mathcal{A}(x) \setminus \mathcal{A}(q)); \\ (x, \delta_m(q,\varepsilon)), & \text{if } e = \varepsilon; \\ \text{Not defined}, & \text{otherwise.} \end{cases}$$

In simple words, given an event $e \in E \cup \{\varepsilon\}$, if e is active both at x and q, and also enabled by the control input $\mathcal{A}(q(q(M+2)))$ in use currently, event e can occur, the plant reaches a new state $x' = \delta(x, e)$, and the supervisor H_M reaches state $q' = \delta_m(q, e)$; if e is only active at x and enabled by the control input $\mathcal{A}(q(q(M+2)))$, then G reaches state x' and H_M remains at q; if $e = \varepsilon$, only the control input is updated to the one in $q' = \delta_m(q, \varepsilon)$; otherwise, the transition is not defined. The synchronization defined above shares the same idea of concurrent composition. We notice that the obtained closed-loop system G_R is an NFA even when G is a DFA due to the random control delays.

Example 3. Fig. 5 illustrates the closed-loop system of G under H_M in Example 2. Clearly, it is equivalent to the DFA in Fig. 3(b). ◇

4. ACTUATOR ENABLEMENT ATTACKS IN NETWORKED SYSTEMS

In the paper, we consider NCSs under attack. Specifically, some controllable events are vulnerable and the attacker

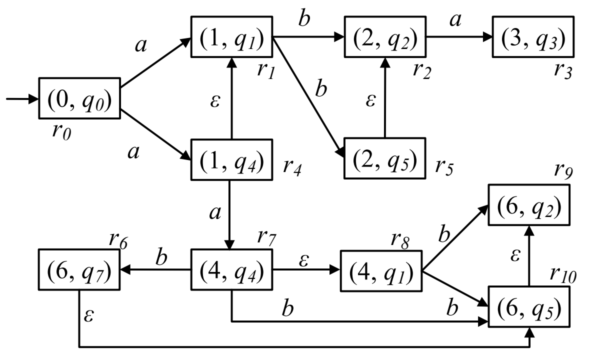


Fig. 5. The closed-loop system G_R of G under H_M.

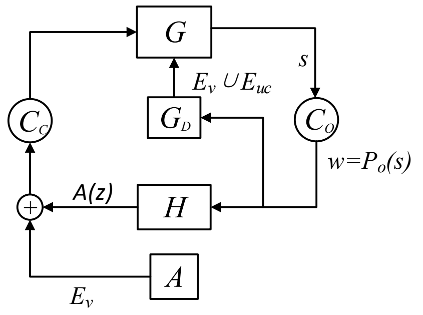


Fig. 6. The closed-loop networked system under attack.

can change the control input on those events from “disable” to “enable” to potentially drive the system to an unsafe state.

4.1 Attack model

The attack model considered is depicted in Fig. 6. The plant G is controlled by the supervisor H. The observation and the control input are transmitted respectively through the observation channel and the control channel in the network. The control input may be delayed and is subject to malicious attacks that can tamper with vulnerable actuators. In particular, the attacker, represented by A, can enable some events disabled by the control input to drive the system towards a pre-defined unsafe state set X_f. The events that may be attacked are called *vulnerable* events and denoted as $E_v \subseteq E_c$. Such an attack is called Actuator Enablement attack (or AE-attack). The attacker is assumed to adopt an “all-out” strategy. In other words, it attacks the vulnerable events at all times. Since the attacker uses “all-out” strategy all the time, it is equivalent to assuming that there is no delay for attacks. Given an arbitrary control input $\mathcal{A}(z)$, the attacked control input is $\mathcal{A}(z) \cup E_v$. Note that due to delays in the control channel, the attacked control input received by the plant may be the one sent from the supervisor $k \leq M$ steps before. To prevent the plant from reaching an unsafe state under attack, there is also an intrusion detection module, called *detector* G_D, that monitors the behavior of the controlled system, and that notifies the plant when the attack is detected. The plant, upon receiving an attack report from the detector, switches to a safer mode where *all* controllable but not vulnerable events[1] are disabled. Note that for security reasons, the communication from the detector to the plant is not via the network. Namely, there is no communication delays.

The goal of the paper is to determine whether the NCS H/G is resilient to AE-attacks, i.e., whether it is possible to detect any attack occurrence and then disable controllable events before the plant reaches an unsafe state. This property is formalized as "AE-safe controllability" in Section 5. In the next section, we first propose an algorithm for constructing the model of the NCS under AE-attacks.

4.2 The model of the closed-loop networked system under AE-attacks

To distinguish the case where events in E_v are attacked by A and not enabled by the supervisor, we denote $E_v^a = \{e^a : e \in E_v\}$ attacker's events and then $E_a = E \cup E_v^a$ the set of all events in the system under attacks.

Algorithm 1 Construction of the attacked closed-loop system

```
Input: Plant $G = (X, E, \delta, x_0)$, and supervisor $H = (Z, E, \delta_h, z_0)$.
Output: NCS under AE-attacks $G_{Ra} = (R, E_a, \delta_{ra}, r_0)$.
1: Build $G_a = (X, E_a, \delta_a, x_0)$, where $\forall x \in X$, $\forall e \in E$,
     $\begin{cases} \delta_a(x,e) := \delta(x,e), & \text{if } e \in \mathcal{A}(x); \\ \delta_a(x,e^a) := \delta(x,e), & \text{if } e \in E_v \cap \mathcal{A}(x). \end{cases}$
2: Build $H_a = (Z, E_a, \delta_{ha}, z_0)$, where $\forall z \in Z$, $\forall e \in E$,
     $\begin{cases} \delta_{ha}(z,e) := \delta_h(z,e), & \text{if } e \in \mathcal{A}(z); \\ \delta_{ha}(z,e^a) := z, & \text{if } e \in E_v \setminus \mathcal{A}(z); \end{cases}$
3: Build $H_{aM} = (Q, E_a, \delta_{am}, q_0)$ of $H_a$
4: $G_{Ra} := G_a \otimes H_{aM}$.
```

The construction of the attacked NCS is summarized in Algorithm 1. First, the automaton G_a is built by adding to each transition $\delta(x,e) = x'$ with $e \in E_v$ a parallel transition $\delta_a(x,e^a) = x'$. At Step 2, based on H we construct the supervisor H_a by adding self-loops of events in $\{e^a | e \in E_v \setminus \mathcal{A}(z)\}$. Namely, the attacker will enable all the vulnerable events that were disabled by the supervisor H. Then following the definition of H_M in Section 3, construct H_{aM} that models H_a with control delays (Step 3). Finally, synchronize G_a and H_{aM} to build the NCS G_{Ra} under AE-attacks (Step 4). Note that events $e^a \in E_v^a$ have the same observability[2] as their corresponding ones in E_c, i.e., $P_o(e^a) = P_o(e)$; Events in E_c are controllable while events in E_v^a are considered as uncontrollable.

The model G_{Ra} describes the dynamics of the NCS under attacks, which can be used for not only attacks detection and prevention but also analysis of other properties of the system under attacks (e.g., opacity).

Example 4. Consider again the NCS in Example 1, and let $E_v = \{c\}$. Adding parallel transitions labeled with c^a to G and self-loops of c^a to H, we construct the automata G_a

[1] Under an attack, vulnerable events become uncontrollable by the supervisor because the attacker can still enable all the vulnerable events in the plant.

[2] The natural projection P_o is extended to $P_o : E_a^* \to E_o^*$.

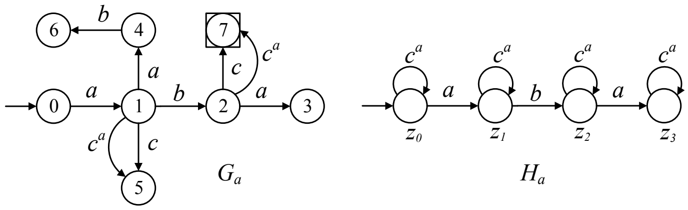


Fig. 7. The plant and the supervisor under attacks.

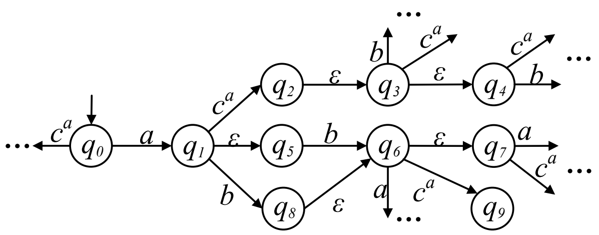


Fig. 8. Part of the model H_{aM} of H_a with control delays.

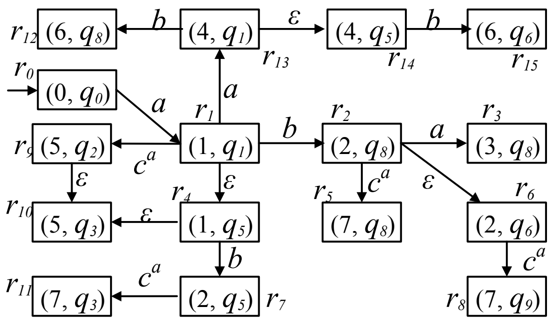


Fig. 9. Part of the closed-loop system G_{Ra} under attacks.

and H_a (in Fig. 7) that include all possible behavior of the plant and the supervisor with attacks. We construct the model H_{aM} with control delays. Fig. 8 shows part of the H_{aM} that will contribute in the synchronization. Finally, part of the closed-loop system under attacks $G_{Ra} = G_a \otimes H_{aM}$ is illustrated in Fig. 9. States r_5, r_8 and r_{11} in G_{Ra} imply that under attacks the plant may reach the unsafe state 7 in several situations: for instance, r_5 represents the case where the plant reaches state 7 because of the attack on $\mathcal{A}(z_2)$ after $s = ab$. ⋄

5. DETECTION AND PREVENTION OF ACTUATOR ENABLEMENT ATTACKS

In the case discussed in Example 4, we can see that with the "all-out" strategy the attacker can lead the plant to unsafe states. However, along the process the fact that the system is under attack may also be detected by the detector G_D when observing a string that does not belong to the specification. Then it may be possible to stop all controllable but not vulnerable events to prevent the plant from reaching unsafe states. In this section, we formalize the notion of *AE-safe controllability* in NCSs. A detection method is proposed, and based on the method AE-safe controllability can also be tested.

5.1 AE-safe controllability

We denote $\Psi(E_v^a) = \{s \in L(G) : s = s'e, s' \in E^*, e \in E_v^a\}$ the set of strings that contain only one attacked event in the end, and $R_f = \{(x,q) \in R : x \in X_f\}$ the set of states in G_{Ra} whose first elements are unsafe states. The notion of AE-safe controllability in NCSs is defined as follows.

Definition 5. System $G_{Ra} = (R, E_a, \delta_{ra}, r_0)$ is said to be *AE-safe controllable* (with respect to P_o, E_v, E_c and X_f) if $(\forall s \in \Psi(E_v^a))(\forall t \in L(G_{Ra})/s)[(\delta_{ra}(r_0, st) \cap R_f \neq \emptyset) \wedge (\forall s' \precneqq st, \delta_{ra}(r_0, s') \cap R_f = \emptyset)] \Rightarrow (t = t_1t_2) \wedge [(\nexists \sigma \in L(G_{Ra}))P_o(st_1) = P_o(\sigma) \wedge E_v^a \nvdash \sigma] \wedge [(E_c \setminus E_v) \vdash t_2]$.

In simple words, the closed-loop system G_{Ra} is AE-safe controllable if, for any string s that has been attacked and for any of its continuations t reaching an unsafe state for the first time, t can be split as $t = t_1t_2$ such that the attack can be detected after st_1, and t_2 contains at least one controllable but not vulnerable event e. Therefore, by disabling e, the evolution st can be stopped before reaching the unsafe state.

Since the attacker is using the "all-out" strategy, once we have proved a system AE-safe controllable, for other "smaller" attack strategy not attacking all the time, the system is also AE-safe controllable.

5.2 Diagnoser for attacks

In Carvalho et al. (2016), it is shown that attack detection, i.e., detecting the occurrence of events in E_v^a, can be reduced to a fault diagnosis problem taking E_v^a as the set of fault events. Note that fault events in $E_v^a \cap \{e^a | e \in E_o\}$ are observable. We denote $E_{ao} = E_o \cup (E_v^a \cap \{e^a|e \in E_o\})$ the set of observable events in G_{Ra}, and its diagnoser[3] $Diag(G_{Ra}) = (Y, E_{ao}, \delta_d, y_0)$, where $Y \subseteq 2^{R\times\{N,F\}}$, $\delta_d : Y \times E_a \to Y$ is the transition function and $y_0 \in Y$ is the initial state. To each state $y = \{(r_1,\gamma_1),(r_2,\gamma_2),\ldots,(r_v,\gamma_v)\}$ of $Diag(G_{Ra})$ we associate a diagnosis value $\varphi(y) \in \{U,T,N\}$ such that:

- $\varphi(y) = N$ (no attack): if $\gamma_i = N$ for all $i \in \{1,2,\ldots,v\}$;
- $\varphi(y) = T$ (attack detected): if $\gamma_i = F$ for all $i \in \{1,2,\ldots,v\}$;
- $\varphi(y) = U$ (uncertain): if $\exists i,j \in \{1,2,\ldots,v\}$ such that $\gamma_i = N$ and $\gamma_j = F$.

Using diagnoser $Diag(G_{Ra})$, AE-safe controllability can be tested by the following proposition.

Proposition 6. Given system $G_{Ra} = (R, E_a, \delta_{ra}, r_0)$ and its diagnoser $Diag(G_{Ra}) = (Y, E_{ao}, \delta_d, y_0)$, G_{Ra} is not AE-safe controllable, if and only if there exists a state $y = \{(r_1,\gamma_1),(r_2,\gamma_2),\ldots,(r_v,\gamma_{jv})\}$ in $Diag(G_{Ra})$ such that one of the following conditions holds:

(1) $\varphi(y) = U \Rightarrow (\exists i \in \{1,\ldots,v\})\ r_i \in R_f$;
(2) $\varphi(y) = T \Rightarrow [(\exists y' \in Y)(\exists e \in E_o)\varphi(y') \in \{U,N\} \wedge \delta_d(y',e) = y] \wedge [(\exists i \in \{1,\ldots,v\})(\exists s \in (E_{uc} \cup E_v^a)^*)\delta_{ra}(r_i,s) \subseteq R_f]$.

Condition (1) implies the plant may reach an unsafe state before the attack is diagnosed. Therefore, the evolution of the plant was not stopped in time. Condition (2) means that for the first time attacks are detected, the plant is able to reach an unsafe state through uncontrollable or vulnerable events. Therefore, even though attacks are

[3] We refer the reader Carvalho et al. (2016) for details on the diagnoser construction.

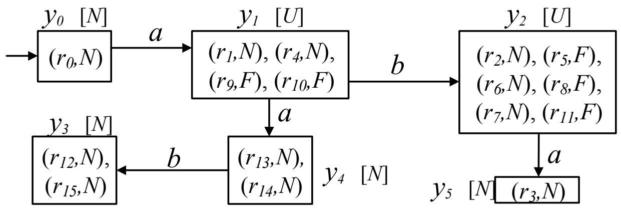

y_0 [N]
(r_0,N)
a
y_1 [U]
(r_1,N), (r_4,N), (r_9,F), (r_{10},F)
b
y_2 [U]
(r_2,N), (r_5,F), (r_6,N), (r_8,F), (r_7,N), (r_{11},F)
a
y_3 [N]
(r_{12},N), (r_{15},N)
b
(r_{13},N), (r_{14},N)
y_4 [N]
a
y_5 [N]
(r_3,N)

Fig. 10. Diagnoser of G_{Ra}.

detected, the plant will be unstoppable from reaching unsafe states. Note that this condition includes the case where $\exists i \in \{1,\ldots,v\}$ such that $r_i \in R_f$. Therefore, if one of the above conditions is satisfied, G_{Ra} is not AE-safe controllable.

Example 7. Let us consider the closed-loop system G_{Ra} in Example 4, where $E_{ao} = \{a,b\}$, $E_{uo} = \{c, c^a\}$ and $R_f = \{r_5, r_8, r_{11}\}$. Its diagnoser is shown in Fig. 10. In $Diag(G_{Ra})$, there is state y_2 such that $\varphi(y_2) = U$ and y_2 contains r_5, r_8, $r_{11} \in R_f$, i.e., y_2 satisfies condition (1) in Proposition 6. Therefore, G_{Ra} is not AE-safe controllable. Indeed, when the detector observes $w = ab$, the string occurred in the NCS may be ab or abc with c being attacked. Thus, the detector is not certain that attacks happened while the plant is already at state 7. $\diamond$

By Proposition 6, to test AE-safe controllability, we need to check uncertain states and the first-meet certain states in the diagnoser. When one of the conditions is satisfied, one can conclude that the system is not AE-safe controllable.

Remark: The detection of attacks follows the same idea of fault diagnosis. Therefore, one can also use verifier instead of diagnoser to detect attacks and test AE-safe controllability, as Carvalho et al. (2018) showing.

6. CONCLUSION AND FUTURE WORK

In the paper, we investigate the problem of attack detection and prevention in NCSs. The plant is observed and controlled by the supervisor via the network, where bounded control delays may occur. An algorithm for constructing the closed-loop system under attacks is proposed and a method for testing AE-safe controllability of the NCS is presented. There are many problems worthy study in the future. For instance, consider other types of attack strategy or prevention mechanism.

REFERENCES

Carvalho, L.K., Wu, Y.C., Kwong, R., and Lafortune, S. (2016). Detection and prevention of actuator enablement attacks in supervisory control systems. In *2016 13th International workshop on discrete event systems (WODES)*, 298–305. IEEE.

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

Lima, P.M., Alves, M.V., Carvalho, L.K., and Moreira, M.V. (2017). Security against network attacks in supervisory control systems. *IFAC-PapersOnLine*, 50(1), 12333–12338.

Lin, F. (2014). Control of networked discrete event systems: Dealing with communication delays and losses.

SIAM Journal on Control and Optimization, 52(2), 1276–1298.

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

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

Tong, Y., Li, Z., Seatzu, C., and Giua, A. (2018). Current-state opacity enforcement in discrete event systems under incomparable observations. *Discrete Event Dynamic Systems*, 28(2), 161–182.

Yin, X. and Lafortune, S. (2015). A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, 61(8), 2140–2154.

Zhang, Q., Seatzu, C., Li, Z., and Giua, A. (2019). Cyber attacks with bounded sensor reading edits for partially-observed discrete event systems.