A cyber attack model with bounded sensor reading alterations

Su, Rong

2017

Su, R. (2017). A cyber attack model with bounded sensor reading alterations. Proceedings of 2017 American Control Conference (ACC), 3200-3205. doi:10.23919/ACC.2017.7963440

<https://hdl.handle.net/10356/138014>

© 2017 AACC. All rights reserved. This paper was published by IEEE in Proceedings of 2017 American Control Conference and is made available with permission of AACC.

Downloaded on 26 Apr 2023 20:46:56 SGT

A Cyber Attack Model with Bounded Sensor Reading Alterations

Rong Su

Abstract—One of the major challenges about cyber physical systems is how to prevent cyber attacks to ensure system integrity. There has been a large number of different types of attacks discussed in the modern control and computer science communities. In this paper we aim to investigate one special type of attacks in the discrete-event system framework, where an attacker can arbitrarily alter sensor readings after intercepting them from a target system in order to trick a given supervisor to issue control commands improperly, driving the system to an undesirable state. We first consider the cyber attack problem from an attacker point of view, and formulate an attack with bounded sensor reading alterations (ABSRA) problem. We then show that the supremal (or least restrictive) ABSRA exists and can be synthesized, as long as the plant model and the supervisor model are regular, i.e., representable by finite-state automata.

Index Terms—discrete-event systems, supervisory control, cyber security, attack under bounded sensor reading alterations, partial observation, controllability

I. INTRODUCTION

A cyber-physical system (CPS) is a mechanism controlled or monitored by computer-based algorithms. Examples of CPS include smart grid, autonomous automobile systems, medical monitoring, process control systems, distributed robotics, and automatic pilot avionics, etc. The connection between the cyber part and the physical part heavily relies on communication networks, which has been raising a major security concern, as different types of cyber attacks can tamper the data collection processes and interfere safety critical decision making processes, which may cause irreparable damages to the physical systems being controlled and to people who depend on those systems.

There has been a growing number of publications addressing the cyber security issues from both the computer science community, which focuses on the computer computation related issues, and the systems control community, which focuses on issues related to the system dynamics affected by cyber attacks. Recently, more and more efforts have been made in classifying different types of malicious attacks, assuming that the attackers are sufficiently intelligent [12] [13], instead of merely just generating random failures, which is well studied in the fields of reliability and fault tolerant control. Typically, an intelligent attacker requires *system knowledge*, and abilities for *resource disclosure* and *resource disruption* in order to

carry out a successful attack, which is covert to a system user until the attacker's goal of causing a damage to the system is achieved. So *covertiness* and *damage infliction* are two major characteristics of a successful attack. By analyzing different intelligent cyber attacks, proper countermeasures may be developed to prevent a target system from being harmed by a specific type of attacks.

In this paper we study a special type of data deception attacks in the discrete-event system framework, where an attacker can intercept sensor measurements (or observations) modeled by observable events and alter them arbitrarily but with an upper bound imposed on the length of each altered observation sequence. By sending those altered observation sequences to a given supervisor, whose function is known to the attacker in advance, the attacker can deliberately and covertly guide the system to move into some undesirable states without making any change to the supervisor. The key challenge is how to “fool” the supervisor to make it believe that the system is operating correctly, while using the supervisor's own control functions to carry out the attack, i.e., to lead the system move into a bad state. To this end, we first propose the concept of *attack under bounded sensor reading alterations* (ABSRA), which can be modelled as a finite-state transducer, possessing the properties of covertness, damage infliction and control feasibility under partial observations. Then we show that the supremal (or least restrictive) ABSRA exists and is computable via a specific synthesis algorithm, as long as both the plant model G and the given supervisor S are finitely representable.

Our construction of an ABSRA model as a transducer is inspired by some recent work on opacity enforcement [14], which aims to use observable event insertions to prevent a potential attacker from correctly determining the actual state of a target system. Due to the different objectives of two works, the modeling details and synthesis algorithms are completely different. There have been some works on cyber attack detection and prevention in the discrete-event community [15] [16] [17], mainly from an adaptive fault tolerant control point of view, which heavily rely on real-time fault diagnosis to identify the existence of an attack and then take necessary robust or adaptive supervisory control actions. In those works the intelligence of an attacker is not considered, and an attack is treated as a fault. As a contrast, we do not rely on real time attack detection, but rely on prior knowledge of attack models.

The remainder of the paper is organized as follows. In

Rong Su is affiliated with Division of Control and Instrumentation, School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798. Emails: rsu@ntu.edu.sg. The support from Singapore Ministry of Education Tier 1 Academic Research Grant M4011221.040 RG84/13 is gratefully acknowledged.

Section II we review the basic concepts and operations of discrete event systems, and formulate an ABSRA synthesis problem, which is then solved in Section III, where we show that the supremal ABSRA exists and computable. A simple yet realistic example runs through the entire paper to illustrate all relevant concepts and algorithms. Conclusions are drawn in Section IV.

II. AN ABSRA PROBLEM

In this section we first recall some standard concepts used in the Ramadge-Wonham supervisory control paradigm. Then we introduce the concept of ABSRA, followed by a concrete ABSRA synthesis algorithm, which reveals that the supremal ABSRA is computable, as long as both the plant model and the given supervisor are regular.

A. Preliminaries on supervisory control

Given an arbitrary finite alphabet Σ , let Σ^* be the free monoid with the empty string ϵ being the unit element and the string concatenation being the monoid operation. Given two strings $s, t \in \Sigma^*$, s is called a *prefix substring* of t , written as $s \leq t$, if there exists $u \in \Sigma^*$ such that $su = t$, where su denotes the concatenation of s and u . Any subset $L \subseteq \Sigma^*$ is called a *language*. The *prefix closure* of L is defined as $\bar{L} = \{s \in \Sigma^* | (\exists t \in L) s \leq t\} \subseteq \Sigma^*$. Given two languages $L, L' \subseteq \Sigma^*$, let $LL' := \{ss' \in \Sigma^* | s \in L \wedge s' \in L'\}$ denote the concatenation of two sets. Let $\Sigma' \subseteq \Sigma$. A mapping $P : \Sigma^* \rightarrow \Sigma'^*$ is called the *natural projection* with respect to (Σ, Σ') , if

- 1) $P(\epsilon) = \epsilon$,
- 2) $(\forall \sigma \in \Sigma) P(\sigma) := \begin{cases} \sigma & \text{if } \sigma \in \Sigma', \\ \epsilon & \text{otherwise,} \end{cases}$
- 3) $(\forall s\sigma \in \Sigma^*) P(s\sigma) = P(s)P(\sigma)$.

Given a language $L \subseteq \Sigma^*$, $P(L) := \{P(s) \in \Sigma'^* | s \in L\}$. The inverse image mapping of P is

$$P^{-1} : 2^{\Sigma'^*} \rightarrow 2^{\Sigma^*} : L \mapsto P^{-1}(L) := \{s \in \Sigma^* | P(s) \in L\}.$$

Given $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, the *synchronous product* of L_1 and L_2 is defined as $L_1 || L_2 := P_1^{-1}(L_1) \cap P_2^{-1}(L_2)$, where $P_1 : (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma_1^*$ and $P_2 : (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma_2^*$ are natural projections. Clearly, $||$ is commutative and associative.

A given target plant is modelled as a *deterministic finite-state automaton*, $G = (X, \Sigma, \xi, x_0, X_m)$, where X stands for the state set, Σ for the alphabet, $\xi : X \times \Sigma \rightarrow X$ for the (partial) transition function, x_0 for the initial state and $X_m \subseteq X$ for the marker state set. We follow the notation system in [10], and use $\xi(x, \sigma)!$ to denote that the transition $\xi(x, \sigma)$ is defined. For each state $x \in X$, let $En_G(x) := \{\sigma \in \Sigma | \xi(x, \sigma)!\}$ be the set of events enabled at x in G . The domain of ξ can be extended to $X \times \Sigma^*$, where $\xi(x, \epsilon) = x$ for all $x \in X$, and $\xi(x, s\sigma) := \xi(\xi(x, s), \sigma)$. The *closed* behavior of G is defined as $L(G) := \{s \in \Sigma^* | \xi(x_0, s)!\}$, and the *marked* behavior of G is $L_m(G) := \{s \in L(G) | \xi(x_0, s) \in X_m\}$. G is *nonblocking* if $\bar{L}_m(G) = L(G)$. We assume that the

marker state set X_m is partitioned into two disjoint sets $X_m = X_{d,m} \dot{\cup} X_{b,m}$, where $X_{d,m}$ is the set of *desirable* states and $X_{b,m}$ denotes the set of *bad* states.

We now recall the concept of supervisors. Let $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc} = \Sigma_o \dot{\cup} \Sigma_{uo}$, where disjoint Σ_c (Σ_o) and Σ_{uc} (Σ_{uo}) denote respectively the sets of *controllable* (*observable*) and *uncontrollable* (*unobservable*) events, respectively. Let $\Gamma := \{\gamma \subseteq \Sigma | \Sigma_{uc} \subseteq \gamma\}$ be the collection of all *control patterns*. A (feasible) *supervisory control map* of G under *partial observation* $P_o : \Sigma^* \rightarrow \Sigma_o^*$ is defined as $V : L(G) \rightarrow \Gamma$, where

$$(\forall s, s' \in L(G)) P_o(s) = P_o(s') \Rightarrow V(s) = V(s').$$

For each $s \in L(G)$, $V(s)$ is interpreted as the set of events allowed to be fired after s . Thus, a supervisory control map will not disable any uncontrollable events, and will impose the same control pattern after strings, which cannot be distinguished based on observations. Let V/G denote the closed-loop system of G under supervision of V , i.e.,

- $\epsilon \in L(V/G)$,
- $L(V/G) := \{s\sigma \in L(G) | s \in L(V/G) \wedge \sigma \in V(s)\}$,
- $L_m(V/G) := L_m(G) \cap L(V/G)$.

The control map V is *finitely representable* if V/G can be denoted by a finite-state automaton, say $S = (Z, \Sigma, \delta, z_o, Z_m = Z)$ such that

- $L(S||G) = L(V/G)$ and $L_m(S||G) = L_m(V/G)$, where $||$ is automaton product [10],
- $(\forall s \in L(S)) E_S(s) := \{\sigma \in \Sigma | s\sigma \in L(S)\} = V(s)$,
- $(\forall s, s' \in L(S)) P_o(s) = P_o(s') \Rightarrow E_S(s) = E_S(s')$.

It has been shown that, as long as a closed-loop language $K \subseteq L_m(G)$ is *controllable* [4] and *observable* [2], there always exists a finitely-representable supervisory control map V such that $L_m(V/G) = K$ and $L(V/G) = \bar{K}$. From now on we assume that V/G is finitely representable by S , which is called a *supervisor*. We assume that S is *legal* in the sense that $L_m(S) \cap L_m(G) \subseteq \{s \in L_m(G) | \xi(x_0, s) \in X_{d,m}\}$, i.e., under the supervision of S , the plant G should never enter any bad marker state.

B. A sensor attack model

We assume that an attacker can intercept each observable event generated by the plant G , and replace it by a sequence of observable events from Σ_o in order to “fool” the given supervisor S , whose function is known to the attacker. Considering that in practice any event occurrence takes an unnegligible amount of time, it is impossible for an attacker to insert an arbitrarily long observable sequence to replace a received observable event. For this reason, we assume that there exists a known natural number $n \in \mathbb{N}$ such that the length of any observable sequence that the attacker can insert is no more than n . Let $\Delta_n := \{s \in \Sigma_o^* | |s| \leq n\}$ be the set of all such bounded observable sequences, where $|s|$ denotes the length of s , and by convention, $|\epsilon| = 0$. We model a sensor attack as a finite state transducer $A = (Y, \Sigma, \Delta_n, \eta, \theta, y_0, Y_m)$, where Y is the state set, Σ

the input alphabet, Δ_n the output alphabets, y_0 the initial state, Y_m the marker state set, which is specifically set as $Y_m = Y$, and $\eta : Y \times \Sigma \times \Delta_n \rightarrow Y$ the (partial) transition map, where for all $\sigma \in \Sigma_{uo}$ and $y \in Y$, $\eta(y, \sigma, \epsilon) = y$, i.e., at each state y all unobservable events are self-looped with ϵ as the output. This is natural because an attacker can only observe observable events, thus, will not make any move upon unobservable events. We still keep unobservable events here to make it easy for us for subsequent technical development. Clearly, $L(A) = L_m(A) \subseteq (\Sigma \times \Delta_n)^*$. Let $\psi : (\Sigma \times \Delta_n)^* \rightarrow \Sigma^*$ and $\theta : (\Sigma \times \Delta_n)^* \rightarrow \Delta_n^*$ be the *input* and *output* maps, respectively, where for each $\mu = (\sigma_1, u_1)(\sigma_2, u_2) \cdots (\sigma_l, u_l) \in (\Sigma \times \Delta_n)^*$, $\psi(\mu) = \sigma_1 \sigma_2 \cdots \sigma_l$ and $\theta(\mu) = u_1 u_2 \cdots u_l$.

The basic procedure of an attack is to intercept every single observable event $\sigma \in \Sigma_o$ generated by the plant G , replace it with some observable string $u \in \Delta_n$, and send u to the supervisor S , in order to trick S to issue a control command $\gamma \in \Gamma$ that may drive the plant G towards a bad marker state. This attack procedure is depicted in Figure 1. The sequential composition of the attack A and

Fig. 1. The block diagram of an attack plan

the supervisor S essentially forms a new supervisor, denoted as $A \circ S$, which receives an observable output $\sigma \in \Sigma_o$ and generates a control command $\gamma \in \Gamma$. The exact definition of this new supervisor reveals the nature of the attack, which is given below. The *sequential composition* of A and S is a deterministic finite state transducer $A \circ S = (Y \times Z \cup \{d\}, \Sigma \times \Delta_n, \zeta, (y_0, z_0), Y_m \times Z_m)$, where d denotes the deadlocking dump state, and for each $(y, z), (y', z') \in Y \times Z$, $(\sigma, u) \in \Sigma \times \Delta_n$, $\zeta(y, z, \sigma, u) = (y', z')$ if one of the following holds,

- $\sigma \in \Sigma_{uo} \wedge u = \epsilon \wedge \eta(y, \sigma, \epsilon) = y' \wedge \delta(z, \sigma) = z'$,
- $\sigma \in \Sigma_o \wedge \eta(y, \sigma, u) = y' \wedge \delta(z, u) = z'$.

For each $(y, z) \in Y \times Z$, $(\sigma, u) \in \Sigma \times \Delta_n$, $\zeta(y, z, \sigma, u) = d$ if $\eta(y, \sigma, u)!$ but $\delta(z, u)$ is undefined. Thus, all transitions that go to the dump state d may potentially reveal the attack, which, for an intelligent attack, should be avoided.

The *impact* of A on the closed-loop system (G, S) is captured by the composition of the plant G and the new supervisor $A \circ S$, i.e.,

$$G \times (A \circ S) = (X \times (Y \times Z \cup \{d\}), \Sigma \times \Delta_n, \kappa = \xi \times \zeta, (x_0, y_0, z_0), X_m \times Y \times Z),$$

where for each $(x, w), (x', w') \in X \times (Y \times Z \cup \{d\})$, $(\sigma, u) \in \Sigma \times \Delta_n$, $(x', w') \in \kappa(x, w, \sigma, u)$ if $x' = \xi(x, \sigma)$ and $w' = \zeta(w, \sigma, u)$. Clearly, $G \times (A \circ S)$ is also a transducer, and it is not difficult to check that

$$G \times (A \circ S) = G \times (\text{Prefix}(G \times (A \circ S)) \circ S),$$

where “=” is in the sense of DES-isomorphism, and $\text{Prefix}(\cdot)$ denotes a function mapping one transducer to another transducer by simply marking every state. In other words, if A is an attack model for the system (G, S) , then $\hat{A} := \text{Prefix}(G \times (A \circ S))$ is also an attack model, which has the same attack effect as that of A on (G, S) . Since $\psi(L(\hat{A})) \subseteq L(G)$, we call $\hat{A}$ a *canonical attack* with respect to (G, S) . Since for any attack, there exists a canonical attack, which has the same attack effect, from now on we only focus on canonical attacks. On the other hand, we will see that A usually is structurally simpler than its canonical one $\hat{A}$, whereas the latter is easier to compute. An interesting question is how to synthesize a simplified attack model A from a given canonical attack model $\hat{A}$, which bears some similarity to the problem of supervisor reduction [8], and will be addressed in our future works.

To illustrate the aforementioned concepts, let us go through a simple single-tank example depicted in Figure 2, which consists of one water supply source whose supply

Fig. 2. A single tank system

rate is q_i , one tank, and one control valve at the bottom of the tank controlling the outgoing flow rate q_o , whose value depends on the valve opening and the water level h . We assume that the valve can only be fully open or fully closed to simplify our illustration, and in case of a full opening, the water level h can only go down. The water level h can be measured, whose value can trigger some predefined events, denoting the water levels: *low* ($h=L$), *medium* ($h=M$), *high* ($h=H$), and *extremely high* ($h=EH$). We construct a simple discrete-event model of the system depicted in Figure 3, where the alphabet Σ contains all events shown in the figure. All events are observable, i.e., $\Sigma_o = \Sigma$. Only the actions of opening the valve ($q_o = 1$) and closing the valve ($q_o = 0$) are controllable, and all water level events are uncontrollable. In the model we use a shaded oval to denote a marker state, i.e., state 5 and state 9 in Figure 3. Assume that we do not want the water level to be extremely high, i.e., the event $h=EH$ should not occur. Thus, state 9 is a bad

Fig. 3. Automaton model of the plant G

marker state, i.e., $X_{d,m} = \{5\}$ and $X_{b,m} = \{9\}$. To prevent state 9 from being reached, we compose a requirement E shown in Figure 4, whose alphabet is $\{h=L, h=M, h=H, h=EH\}$, but the event $h=EH$ is never allowed in the model. A supervisor S can be synthesized by using the standard Ramadge-Wonham supervisory control paradigm, which is also depicted in Figure 4. It is clear that the supervisor S

Requirement E

Supervisor S

Fig. 4. Automaton models of a requirement E (Left) and the supervisor S (Right)

only opens the valve when the water level is high, i.e., it disables the event $q_o = 0$ at state 6 when the event $h=H$ occurs. Our intuition tells us that if an attack always change events of $h=M, h=H, h=EH$ to the event $h=L$, then the supervisor will not prevent the water level from reaching the extreme high level, i.e., the event $h=EH$ will happen. For this reason, we conjecture an attack model A shown in Figure 5, where water levels will be altered to $h=L$, whereas all other events will remain unchanged. The sequential composition $A \circ S$ indicates that, no matter which water level is reached, the attack A always sends $h=L$ to the supervisor S , which tricks it to believe that it is safe to allow the valve to be either closed or opened. The impact of A on the closed-loop system (G, S) is depicted in Figure 6. By marking every state in $G \times (A \circ S)$ we obtain a canonical attack model $\text{Prefix}(G \times (A \circ S))$.

Proposition 1: (1) $\theta(L_m(A \circ S)) \subseteq L_m(S)$; (2) $L_m(A \circ S) \subseteq L_m(A)$; (3) $\psi(L_m(G \times (A \circ S))) = \psi(L_m(A \circ S)) \cap L_m(G)$, $\psi(L(G \times (A \circ S))) = \psi(L(A \circ S)) \cap L(G)$. $\square$

Proposition 2: Given two attacks A_1 and A_2 with the same input alphabet Σ and output alphabet Δ_n , assume that $L(A_1) \subseteq L(A_2)$. Then we have $L(A_1 \circ S) \subseteq L(A_2 \circ S)$. $\square$

Fig. 5. Automaton models of an attack A (Left) and the sequential composition $A \circ S$ (Right)

Fig. 6. Automaton models of $G \times (A \circ S)$

Given two attacks A_1 and A_2 with the same input alphabet Σ and output alphabet Δ_n , let $A_1 \cup A_2$ be their union, which is a deterministic finite-state transducers. Then by the definition of transducer union, we have $L(A_1 \cup A_2) = L(A_1) \cup L(A_2)$.

Proposition 3: $L((A_1 \cup A_2) \circ S) = L(A_1 \circ S) \cup L(A_2 \circ S)$. $\square$

So far we have introduced a simple sensor attack model, and explained how this attack affects the closed-loop system. But we have not described what kind of sensor attacks can be considered intelligent. Next, we will introduce the concept of ABSRA.

C. An ABSRA model

Let $P_o : \Sigma^* \rightarrow \Sigma_o^*$ be the natural projection. An intelligent canonical attack needs to possess the following properties:

- 1) Its insertions must be covert to the given supervisor, i.e.,

$$\theta(L(A)) \subseteq L(S), \quad (1)$$

namely the supervisor will not see any unexpected observable sequences from the attack.

- 2) Any of its insertion sequence may potentially cause damages to G , i.e.,

$$\psi(L(G \times (A \circ S))) = \overline{\psi(L(A \circ S)) \cap \{s \in L_m(G) | \xi(x_0, s) \in X_{b,m}\}}, \quad (2)$$

namely any sequence of insertions by the attack will cause G to reach some bad state eventually. A weaker

version of this property is described below:

$$\psi(L(G \times (A \circ S))) \cap \{s \in L_m(G) \mid \xi(x_0, s) \in X_{b,m}\} \neq \emptyset,$$

which says that the attack A will tamper the absolute correctness of the supervisor S so that there exists some possibility that the system may reach some bad marker state.

- 3) $A \circ S$ forms a standard supervisor for the plant G that enforces normality [2], i.e.,

$$P_o^{-1}(P_o(\psi(L(G \times (A \circ S)))) \cap L(G) = \psi(L(G \times (A \circ S))), \quad (3)$$

and for all $\mu \in L(G \times (A \circ S))$,

$$\psi(En_{G \times (A \circ S)}(\kappa(x_0, y_0, z_0, \mu))) = En_S(\delta(z_0, \theta(\mu)), \quad (4)$$

which denotes that at each state the attack will not intervene the event enablement by the supervisor because we consider only sensor attacks, not actuator attacks.

We call a nonempty model A satisfying the aforementioned four properties (1)-(4) an *Attack with Bounded Sensor Reading Alterations (ABSRA)* of (G, S) .

It is not difficult to check that the attack A shown in Figure 5 does not satisfy Property (1) because S cannot fire $q_o = 0$ before $h=L$, but A can. Nevertheless, the sequential composition $A \circ S$ satisfies all three properties, thus, is an ABSRA. By the aforementioned discussions, we know that the canonical attack model $\text{Prefix}(G \times (A \circ S))$ is also an ABSRA.

Theorem 1: Given a plant G and a legal supervisor S , let $\{A_i \mid i \in I\}$ be a (possibly infinite) collection of ABSRA's with respect to (G, S) . Then $\cup_{i \in I} A_i$ satisfies properties (1)-(4). $\square$

Theorem 1 only implies that the least restrictive (or supremal) attack language exists. But it is not clear whether this supremal language is regular, i.e., whether it can be recognized by a finite-state transducer. Therefore, at this moment the existence of the supremal ABSRA is still unknown. We now state our main problem in this paper.

Problem 1: Given a plant G and a legal supervisor S , design an ABSRA A . $\square$

In the next section we will show that the supremal attack language is regular, i.e., indeed the supremal ABSRA exists, and is computable.

III. SYNTHESIS OF AN ABSRA

We first recall the concepts of controllability [4], and normality [2]. Because we deal with both finite-state automata and finite-state transducers, to make notations simple, we introduce a general purpose alphabet Λ , which can be either $\Lambda = \Sigma$ or $\Lambda = \Sigma \times \Delta_n$, depending on a specific application context. Let $\Lambda_{uc} \subseteq \Lambda$ and $\Lambda_o \subseteq \Lambda$ be an uncontrollable alphabet and an observable alphabet

respectively, where if $\Lambda = \Sigma \times \Delta_n$ then $\Lambda_o := \Sigma_o \times \Delta_n$. Let $\hat{P}_o : \Lambda^* \rightarrow \Lambda_o^*$ be the natural projection. In case that $\Lambda = \Sigma \times \Delta_n$, we have $\hat{P}_o(\epsilon) = \epsilon$, $\hat{P}_o(\sigma, u) = (\sigma, u)$ if $\sigma \in \Sigma_o$, or ϵ otherwise, and $\hat{P}_o(\mu(\sigma, u)) = \hat{P}_o(\mu)\hat{P}_o(\sigma, u)$. When we mention a finite-state transitional structure $\mathcal{G}$, we mean that $\mathcal{G}$ is either a finite-state automaton or a finite-state transducer.

Definition 1: Given a finite-state transitional structure $\mathcal{G}$, a sublanguage $K \subseteq L_m(\mathcal{G})$ is *controllable* w.r.t. $\mathcal{G}$ and Λ_{uc} , if $\overline{K}\Lambda_{uc} \cap L(\mathcal{G}) \subseteq \overline{K}$. $\square$

Definition 2: Given a finite-state transitional structure $\mathcal{G}$, a sublanguage $K \subseteq L_m(\mathcal{G})$ is *normal* w.r.t. $\mathcal{G}$ and Λ_o , if $\hat{P}_o^{-1}(\hat{P}_o(K)) \cap L(\mathcal{G}) = K$. $\square$

Given a finite-state transitional structure $\mathcal{G}$, whose alphabet is Λ , and a requirement $\mathcal{E} \subseteq \Lambda^*$, let

$$\mathcal{CN}(\mathcal{G}, \mathcal{E}) := \{K \subseteq L_m(\mathcal{G}) \mid K \text{ is controllable w.r.t. } \mathcal{G} \text{ and } \Lambda_{uc} \wedge \overline{K} \text{ is normal w.r.t. } \mathcal{G} \text{ and } \Lambda_o\}.$$

By an argument similar to the one used in [4], we can derive that the supremal controllable and normal sublanguage of $L_m(\mathcal{G})$ exists, denoted as $\sup \mathcal{CN}(\mathcal{G}, \mathcal{E})$, such that for all $K \in \mathcal{CN}(\mathcal{G}, \mathcal{E})$, we have $K \subseteq \sup \mathcal{CN}(\mathcal{G}, \mathcal{E}) \in \mathcal{CN}(\mathcal{G}, \mathcal{E})$.

In our setup, an attack is able to arbitrarily alter an observable event. Thus, each event $(\sigma, u) \in \Sigma \times \Delta_n$ is considered controllable, as the attack can choose not to use this alteration. Under this consideration, the uncontrollable alphabet Λ_{uc} is actually empty. Thus, in the following attack model synthesis, we do not explicitly require controllability. This may sound a bit unusual because we do have an uncontrollable alphabet Σ_{uc} for the plant G - how those uncontrollable events affect the attack model synthesis? If we carefully check the properties of an ABSRA, we can see that Property (4) actually implicitly enforces controllability with respect to Σ_{uc} because it requires the attack not to change the event enablement of the supervisor S at the current state, and since by default the supervisor S ensures controllability with respect to Σ_{uc} , and so does the attack model.

Assume that there exists $\Sigma_{o,p} \subseteq \Sigma_o$, which denotes a set of *protected* observable events that cannot be altered by an ABSRA, i.e., given an attack model $A = (Y, \Sigma, \Delta_n, \eta, \theta, y_0, Y_m)$, for all $y \in Y$, $(\sigma, u) \in \Sigma_{o,p} \times \Delta_n$, we have that $\eta(y, \sigma, u)! \Rightarrow u = \sigma$. We now undertake the following ABSRA synthesis procedure.

Procedure 1: (ABSRA Synthesis)

- 1) Input: a plant $G = (X, \Sigma, \xi, x_0, X_m)$, a supervisor $S = (Z, \Sigma, \delta, z_0, Z)$ and $\Sigma_{o,p}$.
- 2) Construct a single-state transducer $A_0 = (Y, \Sigma, \Delta_n, \eta, \theta, y_0, Y)$, where $Y = \{y_0\}$ and the transition map η encodes transitions labeled by a subset of $(\Sigma_o \setminus \Sigma_{o,p}) \times \Delta_n \cup \Sigma_{u,p} \times \Sigma_{u,p} \cup \Sigma_{uo} \times \{\epsilon\}$,

denoting all observable event alterations that the attack wants to consider.

- 3) Let $\mathcal{E}_0 := \{\mu \in L_m(G \times (A_0 \circ S)) \mid \xi(x_0, \psi(\mu)) \in X_{b,m}\}$ be a requirement.
- 4) Undertake the following iteration on $k = 1, \dots$
 - a) Compute $K_k := \sup \mathcal{CN}(G \times (A_0 \circ S), \mathcal{E}_{k-1})$.
 - b) Check property (4) in the definition of ABSRA. If it holds, then go to Step 5). Otherwise, set $\mathcal{E}_k := \{\mu \in K_k \mid \psi(En_{G \times (A_0 \circ S)}(\kappa(x_0, y_0, z_0, \mu))) = En_S(\delta(z_0, \theta(\mu)))\}$, continue the iteration on k .
- 5) Output: A_* , which recognizes $\overline{K_{k+1}}$. $\square$

Theorem 2: Procedure 1 terminates and A_* obtained in Procedure 1 is the supremal ABSRA of (G, S) . $\square$

As an illustration, we apply Procedure 1 to the plant G shown in Figure 3 and the supervisor S shown in Figure 4. We can see that the sensor attack model A in Figure 5 is actually A_0 in Procedure 1 because all events in the model are observable. The composition $A_0 \circ S = A \circ S$ is shown in Figure 5. The outcome of $G \times (A_0 \circ S)$ is shown in Figure 6, which is isomorphic to G . This is not surprising because any string in $L(G)$ may be potentially extended to the bad marker state. The requirement $\mathcal{E}$ is simply the same as $G \times (A_0 \circ S)$. Clearly, we know that K_k is recognizable by a transducer shown in Figure 7, which is almost the same as $G \times (A_0 \circ S)$, except that the only marker state is that bad marker state due to the requirement $\mathcal{E}$. Since all events are

Fig. 7. A transducer recognizing K_*

observable, from $\hat{P}_o(L(A_*)) = \hat{P}_o(\overline{K_*})$ we can derive that $L(A_*) = \overline{K_*}$. Thus, A_* can be chosen by marking every state in $G \times (A_0 \circ S)$, i.e., $A_* = \text{Prefix}(G \times (A_0 \circ S))$, which means A_* is actually a canonical attack of A_0 with respect to (G, S) . By Theorem 2 we know that A_* is the supremal ABSRA of (G, S) .

IV. CONCLUSIONS

In this paper we have first introduced the concept of ABSRA, upon which we have shown that the supremal ABSRA exists and computable, as long as the plant model G and the supervisor S are finitely representable, i.e., their languages are regular. It is interesting to point out that, if we replace the third property of an ABSRA model with a weaker observability property, e.g., the standard observability [2], the supremal ABSRA may not exist any more. Nevertheless, the existence of an ABSRA is

still decidable and computable (with possibly a higher computational complexity), as this ABSRA synthesis problem is equivalent to a synthesis problem of centralized supervisory control under partial observation, which has been shown solvable [9]. Fortunately, the normality property can be easily satisfied in reality, as it only requires that only observable and controllable events can be disabled in online applications - in real industrial applications, it is typical that all control commands are observable.

Acknowledgement

The idea of this paper was originated from a discussion between the author and Prof Stephane Lafortune on opacity enforcement. Without such an inspiring discussion, this paper would never be formed. For this reason, the author would like to thank Prof Lafortune for his contribution.

REFERENCES

- [1] C. Cassandra and S. Lafortune. *Introduction to Discrete Event Systems* (2nd Ed.), Springer, 2008.
- [2] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3):173-198, 1988.
- [3] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.
- [4] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event systems. *SIAM J. Control and Optimization*, 25(1):206-230, 1987.
- [5] R. Su. Discrete-event modeling of multi-agent systems with broadcasting-based parallel composition. *Automatica*, 49(11):3502-3506, 2013.
- [6] R. Su, J.H. van Schuppen and J.E. Rooda. Aggregative synthesis of distributed supervisors based on automaton abstraction. *IEEE Trans. Automatic Control*, 55(7):1627-1640, 2010.
- [7] R. Su, J.H. van Schuppen, J.E. Rooda. Maximally permissive coordinated distributed supervisory control of nondeterministic discrete-event systems. *Automatica*, 48(7):1237-1247, 2012.
- [8] R. Su, W. M. Wonham. Supervisor reduction for discrete-event systems. *Journal of Discrete Event Dynamic Systems*, 14(1):31-53, 2004.
- [9] T.S. Yoo and S. Lafortune. Solvability of centralized supervisory control under partial observation. *Discrete Event Dynamic Systems: Theory and Applications*, 16(4):527-553, 2006.
- [10] W. M. Wonham. *Supervisory Control of Discrete-Event Systems*. Systems Control Group, Dept. of ECE, University of Toronto. URL: www.control.utoronto.ca/DES, 2014.
- [11] W.M. Wonham and P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637-659, 1987.
- [12] A. A. Cardenas, S. Amin, and S. Sastry. Secure control: towards survivable cyber-physical systems. In *Proc. 28th International Conference on Distributed Computing Systems Workshops*, 2008, pp. 495500.
- [13] A. Teixeira, D. Perez, H. Sandberg, and K. H. Johansson. Attack models and scenarios for networked control systems. In *Proc. 1st International Conference on High Confidence Networked Systems*, 2012, pp. 5564.
- [14] Y. C. Wu and S. Lafortune. Synthesis of insertion functions for enforcement of opacity security properties. *Automatica*, 50(5):1336-1348, 2014.
- [15] A. Paoli, M. Sartini and S. Lafortune. Active fault tolerant control of discrete event systems using online diagnostics. *Automatica*, 47(4):639-649, 2011.
- [16] D. Thorsley and D. Teneketzis. Intrusion detection in controlled discrete event systems. In *Proc. 45th IEEE Conference on Decision and Control*, 2006, pp. 60476054.
- [17] L. K. Carvalho, Y. Wu, R. Kwong and S. Lafortune. Detection and prevention of actuator enablement attacks in supervisory control systems. In *Proc. 13th International Workshop on Discrete Event Systems*, 2016, pp. 298305.