2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Supervisor Fortification Against Covert Actuator Attacks*

Ruochen Tai[1], Liyong Lin[1] and Rong Su[1]

Abstract— **This work considers the supervisor fortification problem against covert actuator attacks. A supervisor S' is said to fortify the supervisor S, the latter of which is non-resilient against covert actuator attacks, if S' satisfies two conditions: 1) any covert actuator attack cannot cause damage infliction against S', and 2) S' is control equivalent to S. The key result of this work is that we show the problem of determining the existence of a fortified supervisor to defend against any covert actuator attack, an "exist-for all" decidability question, is decidable. To show the decidability result, we provide a complete and sound procedure that ensures to synthesize a fortified supervisor as long as there exists one.**

I. INTRODUCTION

In the event-driven automated systems, a large amount of works have been done to address the supervisor synthesis problem [1]. However, the network has become an absolutely necessary ingredient nowadays, which might be compromised and used by malicious attacks to inflict damage. In such a cyber-threat environment, the supervisor synthesized without considering the effects of adversaries might not be able to ensure the safe operation anymore. It is then of interest to synthesize a new resilient supervisor that not only can defend against attacks, but also preserve the original closed-loop system behavior, i.e., enforce control equivalence. The requirement of control equivalence is motivated by the observation that we are often asked to design a supervisor to achieve certain performance. In the following text, we call such a resilient and control equivalent supervisor a fortified supervisor. This work centers on the synthesis of fortified supervisors to prevent damage infliction under smart actuator attacks that remain covert without being discovered by particular monitoring mechanisms. Indeed, we shall put efforts on the decidability of whether there exists such a fortified supervisor.

In the discrete-event systems community [2], existing works have proposed many supervisor design strategies against attacks. The strategies of synthesizing supervisors to satisfy a given requirement against a given attack model have been studied in [3]-[13], which are different from our problem because a fortified supervisor is required to defend against all possible *covert* attacks while [3]-[13] only consider *a given attack model*, e.g., the worst-case attack [3]. Note that the the worst-case attack does not care about exposing itself while covert attacks target to cause damage infliction without being detected. Such a difference makes our studied problem more tricky because an infinite number of covert attacks generally exist for a supervisor, making the approaches developed in [3]-[13] ineffective. In [14]-[21], diagnostic methods are designed for the worst-case attacks to disable all the controllable but not vulnerable events when unsafe strings are identified. By comparison, a fortified supervisor does not use a diagnostic tool. In addition, a fortified supervisor defends against covert attacks instead of the worst-case attacks; consequently, disabling all controllable but not vulnerable events is no longer compulsory in our work. The recent work [22] introduces a new strategy of selecting the active supervisor for a given time to defend against sensor attacks. It is clear that the supervisor fortification problem differs from such a supervisor coordination problem.

The works of [23]-[26] are more closely related to this work in terms of the setup. [23]-[25] synthesize resilient supervisors against covert sensor(-actuator) attacks. [26] also studies supervisor fortification against covert actuator attacks. However, none of [23]-[25] has considered the control equivalence. In addition, [23] and [24] only consider sensor attacks that must have the same observations as the supervisor, while our work considers actuator attacks that may have observations different from those of the supervisor. In terms of the technical methods, [23] proposed a heuristic approach, and [25], [26] proposed constraint-based approaches, but all of them are incomplete and cannot solve the decidability issue. [24] solves the decidability issue. However, due to the above-mentioned differences, the technique of [24] fails to solve our studied decidability problem.

In this work, we show the problem of whether there exists a fortified supervisor against covert actuator attacks is decidable, which to our best knowledge has not been solved before. We remark that this is a non-trivial "exist-for all" decision problem due to the following two challenges. Firstly, an infinite number of control equivalent supervisors generally exist. Thus, the approach proposed in [26] of exhaustively verifying the resilience of each control equivalent supervisor is infeasible. Secondly, there can be an infinite number of covert actuator attacks for any supervisor. Consequently, it is infeasible to adopt the approaches proposed in [3]-[5] of reducing this problem to a standard supervisory control problem by treating the composition of the plant and the given attack model as a new plant and the resilient supervisor to be synthesized as a new supervisor. To solve the decidability issue, we first group all the control equivalent supervisors in a finite-state structure, called bipartite behavior-preserving structure, which is a structure of independent interest that is related to supervisor reduction [27]. We remark that All Inclusive Controller (AIC) structure [28], [29] is also a bipartite structure used for supervisor synthesis. However, the behavior-preserving structure constructed in this paper needs to ensure control equivalence, which is not realized in AIC. This difference naturally makes the behavior-preserving structure different from AIC. Then, we perform the chaining

*The research of the project was supported by the Agency for Science, Technology and Research (A*STAR) under its IAF-ICP Programme ICP1900093 and the Schaeffler Hub for Advanced Research at NTU.

[1]The authors are affliated with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. ruochen001@e.ntu.edu.sg; llin5@e.ntu.edu.sg; rsu@ntu.edu.sg.

of three delicately constructed supervisor synthesis procedures to compute a finite-state structure, called bipartite fortified command-nondeterministic supervisor, which encodes all the fortified supervisors. Note that neither the AIC in [28], [29] nor the AIC for Opacity in [30] encodes supervisors satisfying the resilience against covert actuator attacks. We remark that the opacity studied in [30] is defined under the passive intruder that does not influence the dynamics of the closed-loop system. In contrast, the resilience considered in our work is defined under the active actuator attack that could influence the dynamics of the closed-loop system. Thus, our constructed fortified command-nondeterministic supervisor has a different structure from the AIC (for Opacity) in [28]-[30], making the construction techniques presented there not applicable for our work.

There are some other works about actuator attacks (see [19], [31]-[40]). However, they address the problem of covert attacker synthesis, i.e., covert damage string identification, instead of resilient supervisor synthesis. In addition, their approaches also do not work for our problem. Firstly, [40] targets to find all the covert damage strings, where each one works for *all the observation-consistent supervisors*. In contrast, after constructing the behavior-preserving structure in our work, we solve a sub-problem to find all the covert damage strings, where each one works for *at least one control equivalent supervisor*. Thus, the covert damage strings identified in our work differ from those identified in [40], making the approach of [40] no longer applicable. On the other hand, [19], [31]-[39] only find covert damage strings for one given supervisor; thus, their techniques do not apply to identifying covert damage strings in our work, where there can be in general an infinite number of control equivalent supervisors.

This remaining sections are organized as follows. We recap basic notations in Section II. The problem formulation is shown in Section III. The methodology for supervisor fortification problem is introduced in Section IV and Section V. Section VI concludes this work. Due to space limitation, we refer readers to [41] for proof details.

II. PRELIMINARIES

We introduce some preliminaries, mostly following [1], [2]. Σ^* is the Kleene-closure of a finite alphabet Σ. The prefix closure of a language $L \subseteq \Sigma^*$ is defined as $\overline{L} = \{u \in \Sigma^* \mid (\exists v \in L)\, u \leq v\}$. As usual, $P_{\Sigma'} : \Sigma^* \to (\Sigma')^*$ is the natural projection [1]. A finite-state automaton G is given by a 5-tuple $(Q, \Sigma, \xi, q_0, Q_m)$.We write $\xi(q, \sigma)!$ to mean that $\xi(q, \sigma)$ is defined. We define $En_G(q) = \{\sigma \in \Sigma | \xi(q, \sigma)!\}$. Let $L(G)$ and $L_m(G)$ denote the closed-behavior and the marked behavior, respectively. When $Q_m = Q$, we write $G = (Q, \Sigma, \xi, q_0)$ for simplicity. $Ac(G)$ represents the automaton by removing those states (as well as the associated transitions) that are not reachable in G [2]. $G^{|Q-Q'}$ represents the automaton by removing those states (as well as the associated transitions) in $Q' \subseteq Q$. Under the subset $\Sigma' \subseteq \Sigma$, we define the "unobservable reach" of $q \in Q$ to be $UR_{G,\Sigma-\Sigma'}(q) := \{q' \in Q | [\exists s \in (\Sigma - \Sigma')^*]\, q' = \xi(q, s)\}$. $\mathscr{P}_{\Sigma'}(G)$ is defined as the finite-state automaton $(2^Q - \{\varnothing\}, \Sigma, \delta, UR_{G,\Sigma-\Sigma'}(q_0))$ over Σ, where the (partial) transition function $\delta : (2^Q - \{\varnothing\}) \times \Sigma \to (2^Q - \{\varnothing\})$ is defined as follows: 1) for any $\varnothing \neq Q' \subseteq Q$ and any $\sigma \in \Sigma'$, if $\xi(Q', \sigma) \neq \varnothing$, then $\delta(Q', \sigma) = UR_{G,\Sigma-\Sigma'}(\xi(Q', \sigma))$, and 2) for any $\varnothing \neq Q' \subseteq Q$ and any $\sigma \in \Sigma - \Sigma'$, if there exists $q \in Q'$ such that $\xi(q, \sigma)!$, then $\delta(Q', \sigma) = Q'$.

For any two finite-state automata G_1 and G_2, their parallel composition [2] is denoted as $G_1||G_2$. Following [1], [2], for a plant modeled as a finite-state automaton $G = (Q, \Sigma, \xi, q_0, Q_m)$, its event set Σ is partitioned into $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc} = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_c (Σ_o) and Σ_{uc} (Σ_{uo}) are defined as the sets of controllable (observable) and uncontrollable (unobservable) events, respectively. A control constraint over Σ is a tuple (Σ_c, Σ_o). The *Basic Supervisory Control and Observation Problem (BSCOP)* [2] is as follows.

Definition II.1. (BSCOP). Given plant G and legal language $L_a = \overline{L_a}$, find a supervisor S over the control constraint (Σ_c, Σ_o) such that 1) $L(S||G) \subseteq L_a$, and 2) for any other supervisor S' such that $L(S'||G) \subseteq L_a$, $L(S'||G) \subseteq L(S||G)$.

When $\Sigma_c \subseteq \Sigma_o$, the supremal solution always exists for BSCOP [2], although it may be empty.

III. PROBLEM FORMULATION

We first present the system models in the supervisory control architecture under attack. The supervisor fortification problem is then formulated.

A. Component models

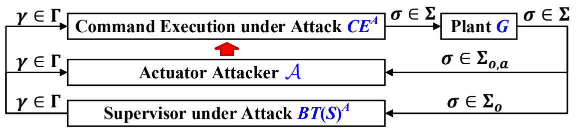
$\gamma \in \Gamma$
Command Execution under Attack CE^A
$\sigma \in \Sigma$
Plant G
$\sigma \in \Sigma$
$\gamma \in \Gamma$
Actuator Attacker $\mathcal{A}$
$\sigma \in \Sigma_{o,a}$
$\gamma \in \Gamma$
Supervisor under Attack $BT(S)^A$
$\sigma \in \Sigma_o$

Fig. 1: System architecture.

Conceptually speaking, we consider four components in the architecture shown in Fig. 1:

- Plant: There are some damage states in the plant, which should be avoided under the control of a supervisor.
- Supervisor under attack: The supervisor issues one control command in $\Gamma = \{\gamma \subseteq \Sigma | \Sigma_{uc} \subseteq \gamma\}$ after the system initiation or upon one observation. Once any unexpected observable sequence is seen, the supervisor asserts that the information inconsistency happens and an attack is detected [23], [24].
- Command execution under attack: It describes the attacked phase from control command reception to event execution at the plant.
- Actuator attacker: The attacker can implement enablement and disablement attacks over the attackable event set $\Sigma_{c,a} \subseteq \Sigma_c$ to distort a control command. The attacker could observe control commands in Γ and (plant) events in $\Sigma_{o,a} \subseteq \Sigma$. We assume that $\Sigma_{c,a} \subseteq \Sigma_{o,a}$. We consider nondeterministic attackers that have more than one attack choice after an observation. Any attacker needs to remain covert against the supervisor.

Remark III.1. Compared with [26] which also studies supervisor fortification, our work considers a more general setup by relaxing the assumptions $\Sigma_c \subseteq \Sigma_o$ and $\Sigma_{o,a} \subseteq \Sigma_o$. As we shall see later, the assumption $\Sigma_{c,a} \subseteq \Sigma_{o,a}$ helps us to establish the decidability result.

Next, we briefly introduce how to model these components as finite-state automata.

1) Plant: It is modeled by $G = (Q, \Sigma, \xi, q^{init}, Q_d)$. The set of damage states is denoted by Q_d.

2) Supervisor: For the supervisor[1] $S = (Q_s, \Sigma, \xi_s, q_s^{init})$, we transform S into the bipartite supervisor where the command sending is explicitly shown. For convenience, we define $\Gamma(q) := En_S(q) = \{\sigma \in \Sigma | \xi_s(q, \sigma)!\} \in \Gamma$, which is the command sent at state q. We denote the bipartite supervisor by $BT(S) = (Q_{bs}, \Sigma_{bs}, \xi_{bs}, q_{bs}^{init})$, where $Q_{bs} = Q_s \cup Q_s^{com} = Q_s \cup \{q^{com} \mid q \in Q_s\}$, $q_{bs}^{init} = (q_s^{init})^{com}$, $\Sigma_{bs} = \Sigma \cup \Gamma$, and ξ_{bs} is defined as follows:

1. $(\forall q^{com} \in Q_s^{com})\, \xi_{bs}(q^{com}, \Gamma(q)) = q$
2. $(\forall q \in Q_s)(\forall \sigma \in \Sigma_{uo})\, \xi_s(q, \sigma)! \Rightarrow \xi_{bs}(q, \sigma) = q$
3. $(\forall q \in Q_s)(\forall \sigma \in \Sigma_o)\, \xi_s(q, \sigma)! \Rightarrow \xi_{bs}(q, \sigma) = (\xi_s(q, \sigma))^{com}$

In the state set Q_{bs}, any $q^{com} \in Q_s^{com}$ is a control state, at which $\Gamma(q)$ is sent, and any $q \in Q_s$ is a reaction state, at which any event in $\Gamma(q)$ may be observed. Thus, the initial state is $(q_s^{init})^{com}$. Case 1, Case 2, and Case 3 encode the control command sending, unobservable event occurrence, and observable event reception, respectively.

Based on $BT(S)$, we construct the bipartite supervisor under attack $BT(S)^A = (Q_{bs}^a, \Sigma_{bs}^a, \xi_{bs}^a, q_{bs}^{a,init})$, where $Q_{bs}^a = Q_{bs} \cup \{q^{detect}\}$, $q_{bs}^{a,init} = q_{bs}^{init}$, $\Sigma_{bs}^a = \Sigma \cup \Gamma$, and ξ_{bs}^a is defined as follows:

1. $(\forall q, q' \in Q_{bs}^a)(\forall \sigma \in \Sigma \cup \Gamma)\xi_{bs}(q, \sigma) = q' \Rightarrow \xi_{bs}^a(q, \sigma) = q'$
2. $(\forall q \in Q_s)(\forall \sigma \in \Sigma_{c,a} \cap \Sigma_{uo})\neg\xi_{bs}(q, \sigma)! \Rightarrow \xi_{bs}^a(q, \sigma) = q$
3. $(\forall q \in Q_s)(\forall \sigma \in \Sigma_o)\neg\xi_{bs}(q, \sigma)! \Rightarrow \xi_{bs}^a(q, \sigma) = q^{detect}$

We add a new state q^{detect}, corresponding to the situation where the actuator attack is detected. Case 2 models the attack effects on unobservable events and Case 3 encodes the attack detection situation.

3) Command execution automaton: To model the command execution process, we first build the automaton $CE = (Q_{ce}, \Sigma_{ce}, \xi_{ce}, q_{ce}^{init})$ [33], where $Q_{ce} = \{q^\gamma | \gamma \in \Gamma\} \cup \{q_{ce}^{init}\}$, $\Sigma_{ce} = \Gamma \cup \Sigma$, and ξ_{ce} is defined as follows:

1. $(\forall \gamma \in \Gamma)\xi_{ce}(q_{ce}^{init}, \gamma) = q^\gamma$
2. $(\forall \gamma \in \Gamma)(\forall \sigma \in \gamma \cap \Sigma_o)\xi_{ce}(q^\gamma, \sigma) = q_{ce}^{init}$
3. $(\forall \gamma \in \Gamma)(\forall \sigma \in \gamma \cap \Sigma_{uo})\xi_{ce}(q^\gamma, \sigma) = q^\gamma$

Next, based on CE, we build the command execution automaton under attack $CE^A = (Q_{ce}^a, \Sigma_{ce}^a, \xi_{ce}^a, q_{ce}^{a,init})$, where $Q_{ce}^a = Q_{ce}$, $q_{ce}^{a,init} = q_{ce}^{init}$, $\Sigma_{ce}^a = \Gamma \cup \Sigma$, and ξ_{ce}^a is defined as follows:

1. $(\forall q, q' \in Q_{ce}^a)(\forall \sigma \in \Sigma \cup \Gamma)\xi_{ce}(q, \sigma) = q' \Rightarrow \xi_{ce}^a(q, \sigma) = q'$
2. $(\forall \gamma \in \Gamma)(\forall \sigma \in \Sigma_{c,a} \cap \Sigma_o)\neg\xi_{ce}(q^\gamma, \sigma)! \Rightarrow \xi_{ce}^a(q^\gamma, \sigma) = q_{ce}^{a,init}$
3. $(\forall \gamma \in \Gamma)(\forall \sigma \in \Sigma_{c,a} \cap \Sigma_{uo})\neg\xi_{ce}(q^\gamma, \sigma)! \Rightarrow \xi_{ce}^a(q^\gamma, \sigma) = q^\gamma$

Case 2 and Case 3 encode actuator attack effects on the attackable event set $\Sigma_{c,a}$.

4) Actuator attacker: We model the attacker by $\mathcal{A} = (Q_a, \Sigma_a = \Sigma \cup \Gamma, \xi_a, q_a^{init})$. Since only events in $\Sigma_{o,a} \cup \Gamma$ can be observed and only events in $\Sigma_{c,a}$ can be disabled by the attacker, $\mathcal{A}$ should satisfy two conditions: 1) ($\mathcal{A}$-controllability) $(\forall q \in Q_a)(\forall \sigma \in \Sigma_a - \Sigma_{c,a})\xi_a(q, \sigma)!$, and 2) ($\mathcal{A}$-observability) $(\forall q \in Q_a)(\forall \sigma \in \Sigma_a - (\Sigma_{o,a} \cup \Gamma))\xi_a(q, \sigma)! \Rightarrow \xi_a(q, \sigma) = q$.

[1]Any supervisor S satisfies two conditions [42]: 1) (state-controllability) $(\forall q \in Q_s)(\forall \sigma \in \Sigma_{uc})\xi_s(q, \sigma)!$, and 2) (state-observability) $(\forall q \in Q_s)(\forall \sigma \in \Sigma_{uo})\xi_s(q, \sigma)! \Rightarrow \xi_s(q, \sigma) = q$.

We refer to $(\Sigma_{o,a}, \Sigma_{c,a})$ as the attack constraint. Clearly, $\mathcal{A}$ is nondeterministic in terms of making attack decisions as it allows multiple attack choices w.r.t. $\Sigma_{c,a}$ upon each observation.

B. Problem formulation

With the above-constructed component models, we model the closed-loop system under attack by $CLS^A = G||CE^A||BT(S)^A||\mathcal{A} = (Q_{cls}^a, \Sigma_{cls}^a, \xi_{cls}^a, q_{cls}^{a,init}, Q_{cls,m}^a)$.

Definition III.1. (Covertness). Given G and S, $\mathcal{A}$ is covert against S over $(\Sigma_{o,a}, \Sigma_{c,a})$ if CLS^A does not reach any state in $\{(q_g, q_{ce}^a, q_{bs}^a, q_a) \in Q_{cls}^a | q_{bs}^a = q^{detect}\}$.

Remark III.2. For a smart attacker, keeping stealthy at all times is often important since remaining hidden in the system paves the way for repetitive damage infliction in different instances of the system. For example, functional abnormality due to stealthy attackers that causes traffic accidents in different autonomous driving cars may be diagnosed as software bugs instead of attacks. From this point of view, we study the resilient control against smart attacks that remain covert at all times, captured by **Definition III.1**.

Definition III.2. (Damage-reachable). Given G and S, $\mathcal{A}$ is damage-reachable against S over $(\Sigma_{o,a}, \Sigma_{c,a})$ if $L_m(CLS^A) \neq \varnothing$.

Definition III.3. (Covert damage string). Given G, S, and $\mathcal{A}$ that is covert and damage-reachable against S over $(\Sigma_{o,a}, \Sigma_{c,a})$, any $s \in L_m(G||CE^A||BT(S)^A||\mathcal{A})$ is a covert damage string that works for S.

Definition III.4. (Resilience). Given G, S is resilient if there does not exist any $\mathcal{A}$ that is covert and damage-reachable against S over $(\Sigma_{o,a}, \Sigma_{c,a})$.

It is assumed there is an original supervisor, denoted as S_{ori}, which is known and non-resilient.

Definition III.5. (Control equivalence). Given G and S_{ori}, a supervisor S' is control equivalent to S_{ori} if $L(G||S_{ori}) = L(G||S')$.

Definition III.6. (Fortification). Given G and S_{ori}, S' is said to be a fortification for S_{ori} if S' is resilient and control equivalent to S_{ori}.

Problem 1. Given G, S_{ori}, and $(\Sigma_{o,a}, \Sigma_{c,a})$, determine whether there exists a fortification for S_{ori}.

IV. BEHAVIOR-PRESERVING STRUCTURE

In this section, we introduce how to group all the control equivalent supervisors in a finite-state structure.

A. Equivalent behavior computation

Firstly, we compute $G||S_{ori}$, the closed-loop system under S_{ori}. Then we construct the structure (cf. Section II) $B = \mathscr{P}_{\Sigma_o}(G||S_{ori}) = (Q_b, \Sigma_b = \Sigma, \xi_b, q_b^{init})$, which adds self-loops labelled by unobservable events for each state in the observer [2] of $G||S_{ori}$. Based on B, we could find all the feasible control commands w.r.t. each observation, which would be explained later.

B. Feasible control commands completion

Based on B, we construct a bipartite structure. The idea is as follows. After an observation encoded in B, we add all the feasible control commands that could retain the control equivalence. We call this structure the bipartite behavior-preserving structure $BPS = (Q_{bps}, \Sigma_{bps}, \xi_{bps}, q_{bps}^{init})$, where

$Q_{bps} = Q_b \cup Q_b^{com} \cup \{q^{dump}\} = Q_b \cup \{q^{com} | q \in Q_b\} \cup \{q^{dump}\}$, $q_{bps}^{init} = (q_b^{init})^{com}$, $\Sigma_{bps} = \Sigma \cup \Gamma$, and ξ_{bps} is defined as follows:

1. $(\forall q \in Q_b)(\forall \gamma \in \Gamma)\mathcal{C}_1 \wedge \mathcal{C}_2 \Rightarrow \xi_{bps}(q^{com}, \gamma) = q$, where
   i. $\mathcal{C}_1 := En_B(q) \subseteq \gamma$
   ii. $\mathcal{C}_2 := (\forall (q_g, q_s) \in q)En_G(q_g) \cap \gamma \subseteq En_B(q)$
2. $(\forall q \in Q_b)(\forall \sigma \in \Sigma_{uo})\xi_b(q, \sigma)! \Rightarrow \xi_{bps}(q, \sigma) = q$
3. $(\forall q \in Q_b)(\forall \sigma \in \Sigma_o)\xi_b(q, \sigma)! \Rightarrow \xi_{bps}(q, \sigma) = (\xi_b(q, \sigma))^{com}$
4. $(\forall q \in Q_b)(\forall \sigma \in \Sigma_{uo})\neg\xi_b(q, \sigma)! \Rightarrow \xi_{bps}(q, \sigma) = q$
5. $(\forall q \in Q_b)(\forall \sigma \in \Sigma_o)\neg\xi_b(q, \sigma)! \Rightarrow \xi_{bps}(q, \sigma) = q^{dump}$
6. $(\forall \sigma \in \Sigma \cup \Gamma)\xi_{bps}(q^{dump}, \sigma) = q^{dump}$

In the state set Q_{bps}, similar to the bipartite supervisor, any state $q^{com} \in Q_b^{com}$ is a control state and any state $q \in Q_b$ is a reaction state. However, different from the transition definition of a bipartite supervisor, after an observable event $\sigma \in \Sigma_o$ occurs at a reaction state q, BPS transits to a control state $(\xi_b(q, \sigma))^{com}$ (Case 3) if σ is defined at q in B; otherwise, i.e., σ is not defined at q in B, BPS transits to the dump state q^{dump} (Case 5).

In ξ_{bps}, for any control state q^{com}, Case 1 adds feasible control commands that maintain equivalent behavior. Two conditions, $\mathcal{C}_1$ and $\mathcal{C}_2$, should be satisfied for any control command γ added at the control state q^{com}. Firstly, $\mathcal{C}_1$ ensures that any event in $En_B(q)$ could be fired under γ. Secondly, it can be checked that, all the possible plant states for the observation sequence $t \in \Sigma_o^*$ are contained in the state $\xi_b(q_b^{init}, t) = q$. Thus, we need to ensure that all the events that may be executed at any plant state q_g ($(q_g, q_s) \in q$) under γ are still contained in $En_B(q)$, which is formulated by $\mathcal{C}_2$. Case 2 and Case 3 retain the transitions encoded in B. Notice that we are constructing a finite-state structure to group all the control equivalent bipartite supervisors, and recall that the structure of a bipartite supervisor requires that all the events in a control command γ should be defined after γ occurs. Hence, for reaction states, we complete events in Σ which are not defined in B in Case 4 and Case 5, where observable transitions lead to the state q^{dump} and unobservable transitions are self-loops. We define Case 6 to self-loop events in $\Gamma \cup \Sigma$ at the state q^{dump}. Notice that this would not break the control equivalence since those completed observable events leading to the state q^{dump} would not occur at all.

C. Structure refinement

We refine BPS by computing $BPS||CE$ because CE encodes all the bipartite supervisors. We call $BPS||CE$ the bipartite behavior-preserving command-nondeterministic[2] supervisor $BPNS = BPS||CE = (Q_{bpns}, \Sigma_{bpns} = \Sigma \cup \Gamma, \xi_{bpns}, q_{bpns}^{init})$. It can be checked that $Q_{bpns} = ((Q_b \cup \{q^{dump}\}) \times \{q^\gamma | \gamma \in \Gamma\}) \dot\cup ((Q_b^{com} \cup \{q^{dump}\}) \times \{q_{ce}^{init}\})$. For convenience, we call any state in $Q_{bpns}^{rea} := (Q_b \cup \{q^{dump}\}) \times \{q^\gamma | \gamma \in \Gamma\}$ a reaction state and any state in $Q_{bpns}^{com} := (Q_b^{com} \cup \{q^{dump}\}) \times \{q_{ce}^{init}\}$ a control state. By construction, events defined at reaction states belong to Σ, and events defined at control states belong to Γ. We have $Q_{bpns} = Q_{bpns}^{rea} \dot\cup Q_{bpns}^{com}$.

[2]$BPNS$ is deterministic, but command non-deterministic because more than one control command may be defined at some control state.

For convenience, the set of supervisors (satisfying state-controllability and state-observability) is denoted by $\mathscr{S}$, and the set of supervisors control equivalent to S_{ori} is denoted by $\mathscr{S}_e(S_{ori}) := \{S' \in \mathscr{S} | L(G||S_{ori}) = L(G||S')\}$. The following theorem indicates all the control equivalent bipartite supervisors are exactly encoded in $BPNS$.

Theorem IV.1. $\bigcup_{S' \in \mathscr{S}_e(S_{ori})} L(BT(S')) = L(BPNS)$.

V. Synthesis of fortified supervisors

In this section, based on $BPNS$ constructed in Section IV, we synthesize a finite-state structure that encodes all the fortified supervisors, from which we could extract one to show the decidability of the supervisor fortification problem.

A. Covert damage strings identification

Firstly, to maintain control equivalence, any fortified supervisor cannot affect its observations of the plant output. Thus, to synthesize fortified supervisors based on $BPNS$, we could only prune those illegal control commands. This motivates us to first identify all the covert damage strings, where each one works for at least one control equivalent supervisor. However, $BPNS$ does not consider the actuator attack effects. Hence, we need to build the version of $BPNS$ under attack, denoted by $BPNS^A = (Q_{bpns}^a, \Sigma_{bpns}^a, \xi_{bpns}^a, q_{bpns}^{a,init})$, where $Q_{bpns}^a = Q_{bpns} \cup \{q_{bpns}^{detect}\} = Q_{bpns}^{rea} \cup Q_{bpns}^{com} \cup \{q_{bpns}^{detect}\}$, $q_{bpns}^{a,init} = q_{bpns}^{init}$, $\Sigma_{bpns}^a = \Sigma \cup \Gamma$, and ξ_{bpns}^a is defined as follows:

1. $(\forall q, q' \in Q_{bpns}^a)(\forall \sigma \in \Sigma \cup \Gamma)\xi_{bpns}(q, \sigma) = q' \Rightarrow \xi_{bpns}^a(q, \sigma) = q'$
2. $(\forall q \in Q_{bpns}^{rea})(\forall \sigma \in \Sigma_{c,a} \cap \Sigma_{uo})\neg\xi_{bpns}(q, \sigma)! \Rightarrow \xi_{bpns}^a(q, \sigma) = q$
3. $(\forall q \in Q_{bpns}^{rea})(\forall \sigma \in \Sigma_o)\neg\xi_{bpns}(q, \sigma)! \Rightarrow \xi_{bpns}^a(q, \sigma) = q_{bpns}^{detect}$

The construction of $BPNS^A$ follows the similar process of $BT(S)^A$, which has been introduced in Section III-A.2. When an unexpected observation is received (Case 3), $BPNS^A$ reaches the state q_{bpns}^{detect}. We have the following.

With $BPNS^A$, we are ready to carry out the following synthesis procedure to identify all the desired covert damage strings.

Procedure 1:

1. Input: G, CE^A, $BPNS^A$, and $\mathscr{C}_{ac} = (\Sigma_{c,a}, \Sigma_{o,a} \cup \Gamma)$.
2. Compute $\mathcal{P} = G||CE^A||BPNS^A = (Q_\mathcal{P}, \Sigma_\mathcal{P}, \xi_\mathcal{P}, q_\mathcal{P}^{init}, Q_{\mathcal{P},m})$.
3. Construct $\mathcal{P}_r = \mathcal{P}^{|Q_\mathcal{P} - Q_{bad}}$, where $Q_{bad} = \{(q, q_{ce}^a, q_{bpns}^a) \in Q_\mathcal{P} | q_{bpns}^a = q_{bpns}^{detect}\}$.
4. Solve a BSCOP where the plant is $\mathcal{P}$, the legal language is $L(\mathcal{P}_r)$, and the control constraint is $\mathscr{C}_{ac} = (\Sigma_{c,a}, \Sigma_{o,a} \cup \Gamma)$. The synthesized supremal solution is denoted as $\hat{\mathcal{A}} = (Q_{\hat{a}}, \Sigma_{\hat{a}}, \xi_{\hat{a}}, q_{\hat{a}}^{init})$.
5. Output: $\hat{\mathcal{A}}$.

In Step 2, we generate $\mathcal{P} = G||CE^A||BPNS^A$. Notice that $Q_{\mathcal{P},m} = Q_d \times Q_{ce}^a \times Q_{bpns}^a$. In Step 3, we remove the state set Q_{bad} in $\mathcal{P}$ to construct $\mathcal{P}_r$. Intuitively speaking, any state in Q_{bad} denotes the situation where the covertness is broken.

In Step 4, we construct a BSCOP, where $\mathcal{P}$ is the plant and $L(\mathcal{P}_r)$ is the legal language. Notice that $\Sigma_{c,a} \subseteq \Sigma_{o,a} \cup \Gamma$ for the control constraint $\mathscr{C}_{ac}$ because of the assumption $\Sigma_{c,a} \subseteq \Sigma_{o,a}$. Thus, the supremal solution $\hat{\mathcal{A}}$ exists. For convenience, the set of attackers that are damage-reachable and covert against the supervisor S is denoted as $\mathscr{A}(S)$. The following theorem shows that $L_m(G||CE^A||BPNS^A||\hat{\mathcal{A}})$ encodes all the covert damage strings, where each one works for at least one control equivalent supervisor.

Theorem V.1. $L_m(G||CE^A||BPNS^A||\hat{\mathcal{A}}) = \bigcup\limits_{S' \in \mathscr{S}_e(S)} \bigcup\limits_{\mathcal{A} \in \mathscr{A}(S')} L_m(G||CE^A||BT(S')^A||\mathcal{A})$.

B. Illegal control commands pruning

Since $L_m(G||CE^A||BPNS^A||\hat{\mathcal{A}})$ encodes all the covert damage strings based on **Theorem V.2** and the attacked versions of all the control equivalent supervisors are encoded in $BPNS^A$, we perform the following procedure to carry out the pruning on $BPNS^A$ with the guidance of $L_m(G||CE^A||BPNS^A||\hat{\mathcal{A}})$.

Procedure 2:

1. Input: G, CE^A, $BPNS^A$, $\hat{\mathcal{A}}$, Σ_o and Γ.
2. Compute $\mathcal{P} = G||CE^A||BPNS^A||\hat{\mathcal{A}} = (Q_{\mathcal{P}}, \Sigma_{\mathcal{P}} = \Sigma \cup \Gamma, \xi_{\mathcal{P}}, q_{\mathcal{P}}^{init}, Q_{\mathcal{P},m})$.
3. Construct $\mathcal{P}_r = (Q_{\mathcal{P}_r}, \Sigma_{\mathcal{P}_r}, \xi_{\mathcal{P}_r}, q_{\mathcal{P}_r}^{init})$ based on $\mathcal{P}$, where
   a. $Q_{\mathcal{P}_r} = (Q_{\mathcal{P}} - Q_{\mathcal{P},m}) \cup \{q^{dump}\}$
   b. $\Sigma_{\mathcal{P}_r} = \Sigma \cup \Gamma$
   c. $q_{\mathcal{P}_r}^{init} = q_{\mathcal{P}}^{init}$
   d. $\xi_{\mathcal{P}_r}$ is defined as:
      i. $(\forall q, q' \in Q_{\mathcal{P}} - Q_{\mathcal{P},m})(\forall \sigma \in \Sigma \cup \Gamma)\xi_{\mathcal{P}}(q,\sigma) = q' \Rightarrow \xi_{\mathcal{P}_r}(q,\sigma) = q'$
      ii. $(\forall q \in Q_{\mathcal{P}} - Q_{\mathcal{P},m})(\forall \sigma \in \Sigma \cup \Gamma)\neg\xi_{\mathcal{P}}(q,\sigma)! \Rightarrow \xi_{\mathcal{P}_r}(q,\sigma) = q^{dump}$
      iii. $(\forall \sigma \in \Sigma \cup \Gamma)\xi_{\mathcal{P}_r}(q^{dump},\sigma) = q^{dump}$
4. Solve a BSCOP where the plant is $BPNS^A$, the legal language is $L(\mathcal{P}_r)$, and the control constraint is $(\Gamma, \Sigma_o \cup \Gamma)$. The synthesized supremal solution is denoted as $S_0^A = (Q_{S_0^A}, \Sigma_{S_0^A} = \Sigma \cup \Gamma, \xi_{S_0^A}, q_{S_0^A}^{init})$.
5. Output: S_0^A.

In Step 2, we compute $\mathcal{P} = G||CE^A||BPNS^A||\hat{\mathcal{A}}$. In Step 3.a, we construct $\mathcal{P}_r$ by removing the marker state set of $\mathcal{P}$ and adding a new state q^{dump}. Then, 1) in Step 3.d.i, for any two non-marker states, the transitions between them defined in $\mathcal{P}$ are retained, 2) in Step 3.d.ii, we add undefined events in $\Sigma \cup \Gamma$ at any non-marker state of $\mathcal{P}$, and those added transitions lead to the new state q^{dump}, and 3) in Step 3.d.iii, for the state q^{dump}, we define the events in $\Sigma \cup \Gamma$. We remark that only the strings leading to damage infliction should be forbidden. Thus, Steps 3.d.ii and 3.d.iii are necessary as they ensure all the legal strings are specified in $\mathcal{P}_r$. Notice that $\mathcal{P}_r$ is not a complete automaton because 1) when any state $q \in Q_{\mathcal{P},m}$ is removed, all the transitions attached to this state q are also removed, and 2) the removed transitions are not completed in Steps 3.d.ii and 3.d.iii. In Step 4, we construct a BSCOP, where $BPNS^A$ is the plant and $L(\mathcal{P}_r)$ is the legal language.

C. Fortified supervisor synthesis

Since the final goal is to synthesize the fortified supervisor, we transform S_0^A to the version in the absence of attacks, denoted as $S_0 = (Q_{S_0}, \Sigma_{S_0}, \xi_{S_0}, q_{S_0}^{init})$, where $Q_{S_0} = Q_{S_0^A}$, $q_{S_0}^{init} = q_{S_0^A}^{init}$, $\Sigma_{S_0} = \Sigma \cup \Gamma$, and ξ_{S_0} is defined as follows:

1. $(\forall q, q' \in Q_{S_0})(\forall \gamma \in \Gamma)\xi_{S_0^A}(q,\gamma) = q' \Rightarrow \xi_{S_0}(q,\gamma) = q'$
2. $(\forall q, q' \in Q_{S_0})(\forall \gamma \in \Gamma)(\forall \sigma \in \gamma \cap \Sigma_{uo})\xi_{S_0^A}(q,\gamma) = q' \Rightarrow \xi_{S_0}(q',\sigma) = q'$
3. $(\forall q, q', q'' \in Q_{S_0})(\forall \gamma \in \Gamma)(\forall \sigma \in \gamma \cap \Sigma_o)\xi_{S_0^A}(q,\gamma) = q' \wedge \xi_{S_0^A}(q',\sigma) = q'' \Rightarrow \xi_{S_0}(q',\sigma) = q''$

Case 1 preserves the transitions labelled by control commands in S_0^A. Case 2 and Case 3 eliminate actuator attack effects by only keeping the transitions labelled as events in γ after γ occurs. In the following text, we refer to $Ac(S_0)$ whenever we talk about S_0. It can be checked that S_0 is a bipartite structure. Thus, we artificially divide the state set of S_0 and write $Q_{S_0} = Q_{S_0}^{rea} \dot\cup Q_{S_0}^{com}$, where $Q_{S_0}^{com}$ is the set of control states and $Q_{S_0}^{rea}$ is the set of reaction states. We have the following observations:

1) Any event in Σ is not defined at any state of $Q_{S_0}^{com}$;
2) Any command in Γ is not defined at any state of $Q_{S_0}^{rea}$;
3) For any state of $Q_{S_0}^{com}$, once a command in Γ occurs, S_0 transits to a reaction state;
4) For any state of $Q_{S_0}^{rea}$, once an observable event in Σ_o occurs, S_0 transits to a control state, and any defined unobservable event in Σ_{uo} is self-looped.

Next, we continue to perform the pruning on S_0 by treating the control states of S_0, where any control command is not defined, as bad states. Notice that this step is necessary because the structure of S_0 may not be consistent with that of a bipartite supervisor which requires that an observation must be followed by a control command, as defined in Section III-A. The following iterative synthesis procedure helps us to remove those states causing structure inconsistencies.

Procedure 3:

1. Input: S_0, Σ_o, and Γ.
2. Let $k := 0$.
3. Compute $Q_{k,del} := \{q \in Q_{S_k}^{com} | En_{S_k}(q) = \varnothing\}$.
4. If $Q_{k,del} \neq \varnothing$, then go to Step 5; if $Q_{k,del} = \varnothing$, then denote $FNS := S_k$ and go to Step 8.
5. Construct $S_{k,r} = S_k^{|Q_{S_k} - Q_{k,del}}$.
6. Solve a BSCOP where the plant is S_k, the legal language is $L(S_{k,r})$, and the control constraint is $(\Gamma, \Sigma_o \cup \Gamma)$. The synthesized supremal solution is denoted as $S_{k+1} = (Q_{S_{k+1}}, \Sigma_{S_{k+1}} = \Sigma \cup \Gamma, \xi_{S_{k+1}}, q_{S_{k+1}}^{init})$. We also denote $Q_{S_{k+1}} = Q_{S_{k+1}}^{rea} \dot\cup Q_{S_{k+1}}^{com}$, where $Q_{S_{k+1}}^{rea}$ is the set of reaction states and $Q_{S_{k+1}}^{com}$ is the set of control states[3].
7. Let $k \leftarrow k+1$ and go to Step 3.
8. Output: FNS.

In Step 2, we introduce a counter k and set it to 0. Step 3 - Step 7 carry out the iteration synthesis. In Step 3, for the k-th iteration, we first find in S_k any control state $q \in Q_{S_k}^{com}$ satisfying the condition $En_{S_k}(q) = \varnothing$, i.e., there is

[3]The division rule is the same as that of $Q_{S_0} = Q_{S_0}^{rea} \dot\cup Q_{S_0}^{com}$.

no control command defined. For convenience, we denote by $Q_{k,del}$ the set of those identified control states. As long as $Q_{k,del}$ is not empty, we proceed to Step 5 to construct $S_{k,r}$ by removing $Q_{k,del}$ from S_k, and then proceed to Step 6 to carry out the synthesis where the plant is S_k with $L(S_{k,r})$ being the legal language. The supremal solution of the synthesis always exists as $\Gamma \subseteq \Sigma_o \cup \Gamma$, and we denote it by S_{k+1}. In Step 4, if $Q_{k,del}$ is empty, then we proceed to Step 8 and output $FNS := S_k$ as the final result. We name the output FNS as the bipartite fortified command-nondeterministic supervisor. The following theorem shows that FNS exactly encodes all the fortifications for S_{ori}.

Theorem V.2. $\bigcup_{S' \in \mathscr{S}_f(S_{ori})} L(BT(S')) = L(FNS)$.

Theorem V.3. Problem 1 is decidable.

VI. CONCLUSIONS

This paper answered a decidability question: whether determining the existence of a fortification (for a non-resilient supervisor) against covert actuator attacks is decidable, where the supervisor and attacker may have different observations. To show the decidability result, we propose a sound and complete decision procedure. In future works, there are several interesting directions. One is to generalize the result in this work by considering the defense against covert sensor-actuator attacks, and another is to relax the assumption $\Sigma_{c,a} \subseteq \Sigma_{o,a}$.

REFERENCES

[1] W. M. Wonham and K. Cai, *Supervisory Control of Discrete-Event Systems*. Cham, Switzerland: Springer, 2019.

[2] C. Cassandras and S. Lafortune, *Introduction to discrete event systems*. New York, NY, USA: Springer, 2009.

[3] R. Meira-Goes, H. Marchand, S. Lafortune, "Towards resilient supervisors against sensor deception attacks", *Proc. IEEE 58th Annu. Conf. Decis. Control (CDC)*, pp. 5144-5149, 2019.

[4] R. Meira-Goes, S. Lafortune, H. Marchand, "Synthesis of supervisors robust against sensor deception attacks", *IEEE Trans. Autom. Control*, DOI 10.1109/TAC.2021.3051459, 2021.

[5] Z. Ma, K. Cai, "On resilient supervisory control against indefinite actuator attacks in discrete-event systems", *IEEE Control Systems Letters*, DOI 10.1109/LCSYS.2022.3168926, 2022.

[6] M. Wakaiki, P. Tabuada, J. P. Hespanha, "Supervisory control of discrete-event systems under attacks", *Dynamic Games and Applications*, vol. 9, no. 4, pp. 965–983, 2019.

[7] Y. Wang and M. Pajic, "Supervisory control of discrete event systems in the presence of sensor and actuator attacks", *Proc. IEEE 58th Annu. Conf. Decis. Control (CDC)*, pp. 5350-5355, 2019.

[8] S. Zheng, S. Shu and F. Lin, "Modeling and Control of Discrete Event Systems under Joint Sensor-Actuator Cyber Attacks", *2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE)*, pp. 216-220, 2021.

[9] M. R. C. Alves, P. N. Pena, K. Rudie, "Discrete-event systems subject to unknown sensor attacks". *Discrete Event Dyn. Syst.*, vol. 32, no. 1, pp. 143-158, 2022.

[10] Y. Wang, Y. T. Li, Z. H. Yu, N. Q. Wu and Z. W. Li, "Supervisory control of discrete-event systems under external attacks", *Inf. Sci.*, vol. 562, pp. 398-413, Jul. 2021.

[11] J. Yao, X. Yin and S. Li, "On attack mitigation in supervisory control systems: A tolerant control approach", *Proc. IEEE 59th Annu. Conf. Decis. Control (CDC)*, pp. 4504-4510, Dec. 2020.

[12] P. M. Lima, M. V. S. Alves, L. K. Carvalho and M. V. Moreira, "Security of cyber-physical systems: Design of a security supervisor to thwart attacks", *IEEE Trans. Autom. Sci. Eng.*, May, 2021.

[13] Y. Wang, M. Pajic, "Attack-resilient supervisory control with intermittently secure communication", *Proc. IEEE 58th Annu. Conf. Decis. Control (CDC)*, pp. 2015-2020, Dec. 2019.

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

[15] L. K. Carvalho, Y.-C Wu, R. Kwong and S. Lafortune, "Detection and mitigation of classes of attacks in supervisory control systems", *Automatica*, vol. 97, pp. 121-133, 2018.

[16] P. M. Lima, M. V. S. Alves, L. K. Carvalho and M. V. Moreira, "Security against network attacks in supervisory control systems", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12333-12338, 2017.

[17] P. M. Lima, L. K. Carvalho and M. V. Moreira, "Detectable and undetectable network attack security of cyber-physical systems", *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 179-185, 2018.

[18] P. M. Lima, M. V. S. Alves, L. K. Carvalho and M. V. Moreira, "Security against communication network attacks of cyber-physical systems", *J. Control Autom. Elect. Syst.*, vol. 30, pp. 125-135, 2019.

[19] A. Khoumsi, "Sensor and actuator attacks of cyber-physical systems: A study based on supervisory control of discrete event systems", *International Conference on Systems and Control (ICSC)*, pp. 176-182, 2019.

[20] Z. Wang, R. Meira-Goes, S. Lafortune and R. Kwong, "Mitigation of classes of attacks using a probabilistic discrete event system framework", *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 35-41, 2020.

[21] Y. Li, Y. Tong, and A. Giua. "Detection and prevention of cyber-attacks in networked control systems", *Proc. 17th Int. Workshop Discrete Event Syst.*, 2020, pp. 7–13.

[22] R. Meira-Goes and S. Lafortune, "Moving Target Defense based on Switched Supervisory Control: A New Technique for Mitigating Sensor Deception Attacks", *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 317-323, 2020.

[23] R. Su, "Supervisor synthesis to thwart cyber-attack with bounded sensor reading alterations", *Automatica*, vol. 94, pp. 35-44, 2018.

[24] R. Su, "On decidability of existence of nonblocking supervisors resilient to smart sensor attacks", arXiv: 2009.02626v1, 2020.

[25] L. Lin, Y. Zhu, R. Su, "Towards bounded synthesis of resilient supervisors", *Proc. IEEE 58th Annu. Conf. Decis. Control (CDC)*, pp. 7659-7664, 2019.

[26] Y. Zhu, L. Lin, R. Su, "Supervisor obfuscation against actuator enablement attack", *Proc. Eur. Control Conf.*, pp. 1760-1765, 2019.

[27] R. Su and W. M. Wonham, "Supervisor reduction for discrete-event systems", *Discrete Event Dyn. Syst.*, vol. 14, no. 1, pp. 31-53, 2004.

[28] X. Yin and S. Lafortune, "Synthesis of maximally-permissive supervisors for the range control problem", *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3914-3929, Aug. 2017.

[29] X. Yin and S. Lafortune, "Synthesis of maximally permissive supervisors for partially-observed discrete-event systems", *IEEE Trans. Autom. Control*, vol. 61, no. 5, pp. 1239-1254, 2016.

[30] X. Yin and S. Lafortune, "A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems", *2015 American Control Conference (ACC)*, pp. 377-383, 2015.

[31] L. Lin, Y. Zhu, R. Su, "Synthesis of covert actuator attackers for free", *Discrete Event Dyn. Syst.*, vol. 30, pp. 561–577, 2020.

[32] L. Lin and R. Su, "Synthesis of covert actuator and sensor attackers as supervisor synthesis," *Proc. 15th Int. Workshop Discrete Event Syst.*, pp. 1-6, 2020.

[33] L. Lin, R. Su, "Synthesis of covert actuator and sensor attackers", *Automatica*, vol 130, 109714, 2021.

[34] L. Lin, S. Thuijsman, Y. Zhu, S. Ware, R. Su, M. Reniers, "Synthesis of supremal successful normal actuator attackers on normal supervisors", *American Control Conference*, pp. 5614-5619, 2019.

[35] R. Meira-Goes, E. Kang, R. Kwong and S. Lafortune, "Stealthy deception attacks for cyber-physical systems", *Proc. IEEE 56th Annu. Conf. Decis. Control (CDC)*, pp. 4224-4230, Dec. 2017.

[36] R. Meira-Goes, E. Kang, R. Kwong and S. Lafortune, "Synthesis of sensor deception attacks at the supervisory layer of cyber-physical systems", *Automatica*, vol. 121, 2020.

[37] S. Mohajerani, R. Meira-Goes and S. Lafortune, "Efficient Synthesis of Sensor Deception Attacks Using Observation Equivalence-Based Abstraction", *IFAC-Papers OnLine*, vol. 53, no. 4, pp. 28-34, 2020.

[38] R. Meira-Goes, R. Kwong and S. Lafortune, "Synthesis of optimal multi-objective attack strategies for controlled systems modeled by probabilistic automata", *IEEE Trans. Autom. Control*, vol. 67, no. 6, pp. 2873-2888, 2022.

[39] Q. Zhang, C. Seatzu, Z. Li, and A. Giua, "A framework for the analysis of supervised discrete event systems under attack", *15th European Workshop on Advanced Control and Diagnosis*, pp. 529-546, 2022.

[40] R. Tai, L. Lin, Y. Zhu and R. Su, "Synthesis of the supremal covert attacker against unknown supervisors by using observations", *IEEE Trans. Autom. Control*, vol. 86, no. 6, pp. 3453-3468, 2023.

[41] R. Tai, L. Lin, and R. Su, "Supervisor obfuscation against covert actuator attackers", *arXiv preprint arXiv:2205.02383*, 2022.

[42] A. Bergeron, "A unified approach to control problems in discrete event processes", *RAIRO-Theoretical Informatics and Applications*, vol. 27, no. 6, pp. 555-573, 1993.