




Article

Language Recovery in Discrete-Event Systems against Sensor Deception Attacks

Abdeldjalil Labeled ^{1,†}, Ikram Saadaoui ^{2,†}, Hanyu E ^{3,†} , Mohammed A. El-Meligy ^{4,†} , Zhiwu Li ^{1,*,†} and Mohamed Sharaf ^{4,†} 

¹ Institute of Systems Engineering, Macau University of Science and Technology, Taipa 999078, Macau SAR, China; 1909853omi30002@student.must.edu.mo

² Mediterranean Institute of Technology, South Mediterranean University, Tunis 99628, Tunisia; ikram.saadaoui@medtech.tn

³ Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6R 2V4, Canada; hanyu6@ualberta.ca

⁴ Industrial Engineering Department, College of Engineering, King Saud University, P.O. Box 800, Riyadh 11421, Saudi Arabia; melmeligy@ksu.edu.sa (M.A.E.-M.); mfsharaf@ksu.edu.sa (M.S.)

* Correspondence: zwli@must.edu.mo

† These authors contributed equally to this work.

Abstract: Cyber-physical systems are characterized by the intrinsic combination of software and physical components that usually include (wired and wireless) communication devices, sensors, actuators, and control processing units. Some wireless devices communicate over insecure channels, rendering cyber-physical systems at risk of malicious attacks that might lead to catastrophic damage. This paper touches upon the problem of sensor deception attacks in supervisory control of discrete-event systems, where an attacker can insert, delete, or replace sensor readings to mislead the supervisor and induce system damage. We model potential attacks using nondeterministic finite-state transducers and then introduce a new defence strategy that utilizes insertion functions. Insertion functions are a type of monitoring interface that alters the system's behaviour by adding extra observable events. Finally, we construct a nondeterministic finite-state transducer called a supervisor filter that recovers the original language generated by the plant by handling the altered language. The insertion function and the supervisor filter cooperate to control the system and confuse the intruder without confusing the supervisor.

Keywords: discrete event system; automaton; insertion function; finite-state transducer; cyber-security; sensor deception attack

MSC: 93E99



Citation: Labeled, A.; Saadaoui, I.; E, H.; El-Meligy, M.A.; Li, Z.; Sharaf, M. Language Recovery in Discrete-Event Systems against Sensor Deception Attacks. *Mathematics* **2023**, *11*, 2313. <https://doi.org/10.3390/math11102313>

Academic Editor: Ivan Lorencin

Received: 15 March 2023

Revised: 8 April 2023

Accepted: 13 April 2023

Published: 16 May 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

A cyber-physical system typically consists of a physical process with sensors that report the process's status to a controller, which then sends commands or control signals to actuators to keep the system in its desired states. Cyber-physical systems are an assemblage of contemporary computer-integrated systems, including the infrastructures of human-being society [1]. Recent years have witnessed a suite of studies that tackle cyber-security issues in discrete-event systems [2–7]. Much attention has focused on the attacks on sensors and actuators at the supervisory layer of cyber-physical control systems. In a networked environment, it is assumed that sensors and actuators can be compromised by an external attacker (typically assumed to be malicious) who may alter the sensor readings provided to a supervisor or the control actions sent to actuators. A thorough overview on cyber attacks and defences for the supervisory control of cyber-physical systems is found in Refs. [8–13].

This paper addresses the problem of creating an attack-resilient cyber-physical system under sensor deception attacks. We assume that a cyber-physical system is modelled as a

discrete-event system, with sensor outputs that belong to a finite set of (observable) events, and vulnerable sensor readings that could be tampered by a hostile attacker seeking to damage the system. For example, an automated teller machine (ATM) is vulnerable to sensor deception attacks, where an attacker tampers with a sensor to report false information about the amount of cash in the machine. This can cause the ATM to stop dispensing cash and lead to customer frustration and lost revenue for the bank. An attack-resilient system usually takes the envelope of the worst-case scenario and then designs a supervisor such that the system operates safely, even in the presence of an attack.

Two main strategies have been proposed in the literature to avoid the failure of a supervisory control system under attack. The first strategy is limited to attacks that can be detected or diagnosed, called a risky or non-stealthy attack. It usually incorporates intruder detection modules (IDMs) into an existing supervisor that is assumed to be well-developed for pre-defined control specifications. In this context, security is typically represented in the same way as diagnosability [14–16], where the attacker is modelled as a faulty behaviour and system resilience as fault tolerance [17]. This strategy exploits fault diagnosis tools to check the existence of a supervisor that is capable of performing the predefined control actions adequately under the worst-case attack scenarios including sensor attacks [18,19], actuator attacks [20], and both sensor and actuator attacks [21–23].

The mechanism captured by the first strategy is not enough or sufficient to protect a cyber-physical system from cyber-attacks, since, unlike passive diagnostic methods, the attacker in this case is often strategic and may need to be modelled as a separate supervisor acting on the system. For example, Liu et al. [24] show that fault-detection algorithms in power grids can be bypassed by an adversary that sends false data that is consistent with plausible power grid configurations.

The second strategy is based on the synthesis of a supervisor, which is resilient to a specific attacker, called a non-risky or smart attacker, that only conducts an attack if it remains undetected afterwards or the attack will result in a certain system damage [25]. Several studies in the literature have investigated the problem of robust supervisor synthesis in the presence of non-risky attackers on sensors [2,26–30], on actuators [31–33], and on both of them [5,34,35]. Some works further restrict the attacker by considering a deterministic one [36]. In this context, determinism and covertness limit the attacker's capability as no risk can be taken, and few choices are available, making it harder to attack a system successfully.

In this work, however, we model an attacker as a nondeterministic finite-transducer to account for a broad range of actions on a specific set of compromised sensors. In addition, we consider a possibly risky attacker that may send strings that are not acceptable to the plant's behaviour to the supervisor. We propose a defence strategy that overcomes the limitations of intruder detection modules and robust supervisor synthesis approaches based on an insertion function at run-time that is placed at the output of the system and a filter that is integrated into the existing supervisor. The insertion function and the filter cooperate to confuse the attacker and recover the original language generated by the plant in order to achieve the control goal. Thus, keeping the current supervisor prevents us from starting over with a new supervisor synthesis and from changing the standard definitions of observability and controllability regarding the impact of attacks on the system.

To the best of our knowledge, this is the first work that aims to develop attack-resilient cyber-physical systems using the notion of insertion function. The major contributions of this paper are stated as follows:

- In our problem formulation, we consider a worst-case attack scenario, in which an attacker (maybe risky) with a nondeterministic attack function performs a broad range of attack actions such as insertion, deletion, and replacement attacks;
- We propose a novel defence mechanism based on the use of insertion functions, which is modelled as a finite state transducer. The main idea is to safeguard the system's compromised behaviour by exploiting the safe channels;
- Given an insertion function, we provide an algorithm to construct a nondeterministic finite-state transducer, called a supervisor filter, that recovers the original language by

handling the altered language received from the attacker. The insertion function and the supervisor filter cooperate to control the system and confuse the intruder without confusing the supervisor.

The remainder of this paper is organized as follows. In Section 2, we briefly recall basic notions related to finite state automata and finite-state transducers. In Section 3, we present and identify the main attributes of the sensor deception attack problem. In Section 4, we illustrate how finite-state transducers are used in attack modelling. The defence strategy using the insertion function and the supervisor filter is provided in Section 5. Section 6 presents a case study on vehicle location. Finally, we conclude the paper in Section 7.

2. Preliminaries

In this section, we provide an overview of a transition system represented as a finite-state automaton (FSA), specifically a deterministic finite-state automaton (DFA). A DFA is defined as a five-tuple $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is a finite set of states, Σ is the set of events or alphabet, $\delta : Q \times \Sigma \rightarrow Q$ is the deterministic transition function, q_0 is the initial state, and $Q_m \subseteq Q$ is the set of final or marked states. We denote by $\delta(q, \sigma)!$ a defined transition in G , indicating that event $\sigma \in \Sigma$ is active (or feasible) at state $q \in Q$. Transition function δ can be naturally extended to $\delta : Q \times \Sigma^* \rightarrow Q$ by defining $\delta(q, \varepsilon) = q$ and $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$ for $q \in Q$, $s \in \Sigma^*$, $\sigma \in \Sigma$, and $\varepsilon \in \Sigma^*$ is the empty string. Similarly, the transition function can be also extended to $\delta : 2^Q \times \Sigma \rightarrow 2^Q$.

The behaviour of automaton G , represented by its generated language and denoted by $L(G)$, is formally defined as $L(G) = \{s \in \Sigma^* \mid \delta(q_0, s)!\}$. We define by $A(q) = \{e \in \Sigma \mid \delta(q, e)!\}$ the set of active events at state $q \in Q$ and by $A(X) = \{e \in \Sigma \mid \exists q \in X, \delta(q, e)!\}$ the set of active events at the subset of states $X \subseteq Q$, i.e., $A(X) = \bigcup_{q \in X} A(q)$.

A relation R between two sets I and O is a set of ordered pairs (i, o) where $i \in I$ and $o \in O$, denoted as $R \subseteq I \times O$. Given $i \in I$, the set of all o such that $(i, o) \in R$ is defined as $R(i) = \{o \in O \mid (i, o) \in R\}$. If for any $i \in I$, $|R(i)| \leq 1$, the relation $R(i)$ is a partial function. Additionally, for a subset $I_0 \subseteq I$, we define $R(I_0) = \{o \in O \mid (i_0, o) \in R, i_0 \in I_0\}$, which is a function $2^I \rightarrow 2^O$. The inversion of a relation R is defined as $R^{-1} = \{(o, i) \in O \times I \mid (i, o) \in R\}$. Finally, the composition of two relations $R \subseteq I \times O$ and $R_0 \subseteq I_0 \times O_0$ is defined as $R \circ R_0 = \{(i, o_0) \in I \times O_0 \mid \exists o \in O \cap I_0 : (i, o) \in R \wedge (o, o_0) \in R_0\}$.

Compared with a finite state automaton, a finite-state transducer (FST) augments a transition with an output label in addition to the input label. In this sense, an FST is a finite-state automaton with two memory tapes: an input tape and an output tape, which maps between two sets of symbols. In plain words, an FSA defines a formal language by accepting a particular set of strings, while an FST defines a relation between sets of strings.

Definition 1. A finite state transducer is a six-tuple $T = (Q, I, O, \eta, q_0, Q_m)$, where Q is the set of states, I is a finite set of inputs, O is a finite set of outputs, $\eta : Q \times (I \cup \{\varepsilon\}) \rightarrow (O \cup \{\varepsilon\}) \times Q$ is the (partial) transition function, q_0 the initial state, and $Q_m \subseteq Q$ is a finite set of final states.

Let T be a finite transducer. The input and output languages of T are respectively represented by $L_{in}(T)$ and $L_{out}(T)$, where $L_{in}(T) = R^{-1}(O^*)$ and $L_{out}(T) = R(I^*)$. A transition of T is denoted (q, u, v, q') ; we say that $q \xrightarrow[u]{u} q'$ is a path from q to q' labeled with (u, v) .

Example 1. We consider the plant T in Figure 1, where $Q = \{0, 1\}$, $I = \{a, b\}$, $O = \{x, y\}$, $q_0 = \{0\}$, and $Q_m = \{1\}$. We have $L_{in}(T) = \{a, ab, abb, abbb, \dots\}$. FST T replaces the input symbol a with x and b with y , for instance $\eta(0, a) = (1, x)$. The transducer performs the following mapping:

$$\begin{aligned} 0 &\xrightarrow[x]{ab} 1 \\ 0 &\xrightarrow[xyy]{abb} 1 \\ &\dots \end{aligned}$$

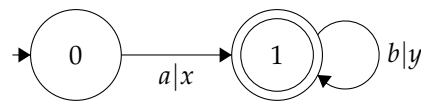


Figure 1. Finite-state transducer T translating words in ab^* to words in xy^* .

3. Problem Formulation

3.1. Assumptions and General Setup

As outlined in the supervisory control theory, originated from Ramadge and Wonham's research in discrete-event systems [37], we examine a networked supervisory control system as depicted in Figure 2. It is assumed that the system, referred to as “plant” G , has a collection of sensors that may be susceptible to vulnerabilities. The behaviour of G is controlled by a “supervisor” S that enforces a safety property on G .

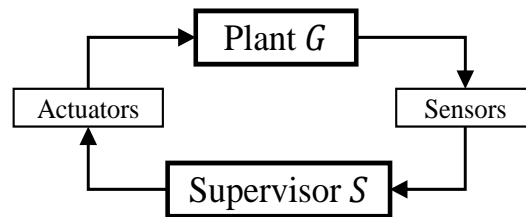


Figure 2. A plant G controlled by a supervisor S .

Communication between the supervisor and the plant, and vice versa, occurs through various channels. Sensor channels are used to communicate information collected by sensors from the plant to the supervisor, while supervisory control channels are used to transfer control actions, such as enabling and disabling the controllable events (actuators), from the supervisor to the plant.

The controlled behaviour is represented by a new discrete event system (DES) denoted by S/G , with a closed-loop language $L(S/G)$ defined as follows: $\varepsilon \in L(S/G)$, i.e., closed behaviour of S/G and, $[s \in L(S/G), \sigma \in S(s), s\sigma \in L(G)] \implies s\sigma \in L(S/G)$. The language $L(G)$ represents the uncontrolled system behaviour, as it includes all possible executions of G .

In addition to the plant and the supervisor, we consider the existence of an attacker A who can perform man-in-the-middle attacks by intercepting the communication channels between the system's sensors and the supervisor, as shown in Figure 3. We assume that the attacker has access to a subset of observable events Σ_o , representing vulnerable sensors, denoted by $\Sigma_{com} \subseteq \Sigma_o$. It also has the capacity to alter some of the sensor readings in these communication channels by injecting false events and removing or replacing authentic ones. Suppose that G has a set of damaging states that could cause physical damage to the plant. The goal of A is to mislead the supervisor and steer G towards a critical state. In this paper, we assume that the supervisor S issues a new control command when the system starts and each time it observes an event $e \in \Sigma_o$. Otherwise, the plant G executes the previous control command, and the supervisor S waits for the next observable event to issue a new one. Similarly, the attacker acts in response to observing a new event $e \in \Sigma_o$ that is executed by G . It is also assumed that the attacker observes the same observable events as S . If the attacker is not present or does not carry out attacks, the system structure is simplified to a basic supervisory control setup.

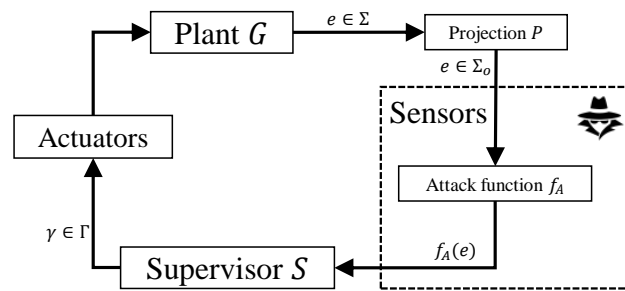


Figure 3. System architecture.

3.2. System Description

We consider a partially observed system that is consistent with standard supervisory control, in which the output symbol for each event is deterministically generated using the natural projection function.

Plant: The plant is abstracted as a *partially-observed deterministic finite state automaton* $G = (Q, \Sigma, \delta, q_0)$, where $\Sigma = \Sigma_o \cup \Sigma_{uo}$ with Σ_o and Σ_{uo} being the sets of observable and unobservable events, respectively. The limited sensing capability of G is illustrated by the natural projection $P_o : \Sigma \rightarrow \Sigma^*$ that is recursively defined as follows:

$$P_o(\varepsilon) = \varepsilon \text{ and } P_o(se) = \begin{cases} P_o(s)e, & \text{if } e \in \Sigma_o \\ P_o(s), & \text{if } e \in \Sigma_{uo} \end{cases}$$

In addition, the restricted actuation capability of G is represented by dividing the event set into $\Sigma = \Sigma_c \cup \Sigma_{uc}$, where Σ_c and Σ_{uc} are the sets of events that are controllable and uncontrollable, respectively. The plant G sends an observable event $e \in \Sigma_o$ to the supervisor S whenever an observable transition $(q, e) = q'$ takes place.

Supervisor: Formally, a supervisor S under partial observation of plant G is a function $S : P_o(L(S/G)) \rightarrow \Gamma$, where $\Gamma = \{\gamma \subseteq \Sigma \mid \Sigma_{uc} \subseteq \gamma\}$ is the set of admissible control decisions. The admissibility of a control decision ensures that uncontrollable events are never disabled. Note that, for all $s \in P_o(L(S/G))$, we have $\Sigma_{uo} \subseteq \Sigma_{uc} \subseteq S(s)$.

Without loss of generality, we assume that S is realized by a finite state automaton $R = (X, \Sigma, \zeta, x_0)$, known as the supervisor realization, that satisfies the controllability and observability constraints as follows:

- (controllability) for any state $x \in X$ and any uncontrollable event $\sigma \in \Sigma_{uc}$, $\zeta(x, \sigma)!$;
- (observability) for any state $x \in X$ and any unobservable event $\sigma \in \Sigma_{uo}$, $\zeta(x, \sigma)!$ implies $\zeta(x, \sigma) = x$.

Note that a partial-observation supervisor's realization catches the current set of enabled events in its active event set; specifically, enabled unobservable events are represented by self-loops at the current state of S .

Attacker: The attacker can alter (i.e., delete, insert, or replace) some sensor readings by performing attacks on compromised sensors. We denote by $\Sigma_{com} \subseteq \Sigma_o$ the set of *compromised events*. Let ch_{si} denote a sensor channel. To keep things simple, we shall refer to a sensor as compromised if its reading is delivered to the supervisor through an attacked channel ch_{si} . In practice, the set of compromised events, denoted by Σ_{com} , is the disjoint union of the observable events that are delivered through the compromised sensor channels, indexed by I . This can be represented as $\Sigma_{com} = \bigcup_{i \in I} \Sigma_{si}$, where Σ_{si} is the set of observable events delivered through the channel ch_{si} . We denote by $\Sigma_{safe} = \Sigma_o \setminus \Sigma_{com}$ the set of safe events that are transmitted through protected channels. An attacker is formally defined as a nondeterministic edit function. Compared with the deterministic model in Ref. [27], the non-deterministic model provides different attack options. The outcome of such a function is selected randomly from a set of possible outcomes when the nondeterministic edit function is implemented.

Definition 2. Let G be a plant with a set of compromised events $\Sigma_{com} \subseteq \Sigma_0$. An attacker is recursively defined by an attack function $f_A : \Sigma_0 \cup \{\varepsilon\} \rightarrow 2^{\Sigma_0^*}$ satisfying

$$(a) f_A(e) \subseteq \begin{cases} \Sigma_{com}^* & \text{if } e \in \Sigma_{com} \cup \{\varepsilon\} \\ \Sigma_{com}^* \{e\} \Sigma_{com}^* & \text{if } e \in \Sigma \setminus \Sigma_{com} \end{cases}$$

$$(b) f_A(se) \subseteq f_A(s)f_A(e) \text{ for all } s \in \Sigma^* \text{ and } e \in \Sigma.$$

Example 2. Consider the networked feedback control structure whose communication network is depicted in Figure 4. Let G be the automaton model of the plant shown in Figure 5a, where $\Sigma = \{a, b, c, d, e\}$, $\Sigma_0 = \{a, b, c, d\}$, $\Sigma_{uo} = \{e\}$, $\Sigma_c = \{a, c, d, e\}$, and $\Sigma_{uc} = \{b\}$. According to Figure 4, the occurrences of events in $\Sigma_{s1} = \{a, d\}$ are transmitted through observation channel ch_{s1} and the occurrences of events in $\Sigma_{s2} = \{b, c\}$ are transmitted through observation channel ch_{s2} . The supervisor shown in Figure 5b guarantees that state 6 is unreachable in the supervised system S/G by satisfying the admissible language $\{aec, dabc\}$, i.e., $L(S/G) = \bar{K}$.

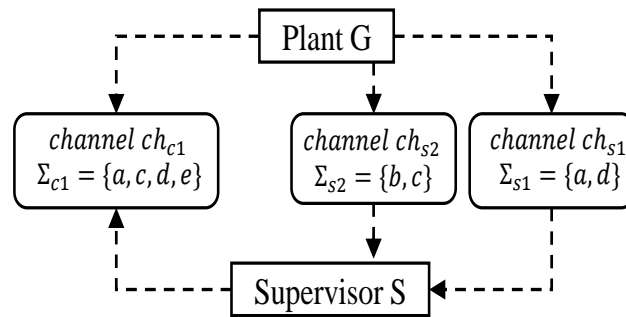


Figure 4. Networked feedback control system.

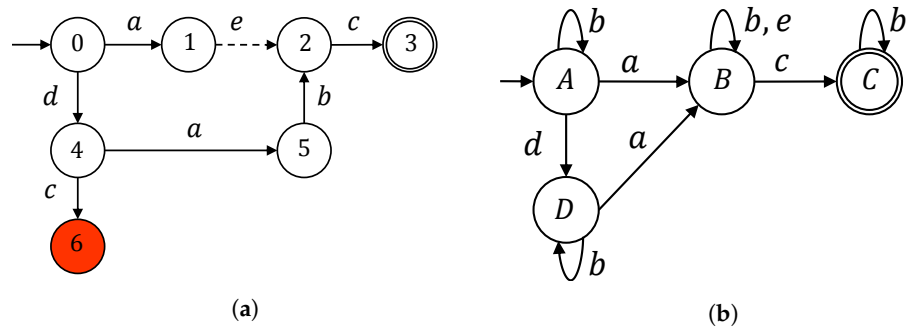


Figure 5. Closed-loop system components. (a) Plant G ; (b) Supervisor realisation S .

4. Sensor Attacks Modeling Using FST

In the field of security, flawed assumptions in the model can provide the simplest means for an attacker to bypass any security measures. As such, we opt to represent sensor attack capabilities through the use of a (partial) finite-state transducer $A = (Q, \Sigma_o, \Sigma_o, \delta, q_0)$. Finite-state transducers can be used to capture a broad range of attacks such as insertion, deletion, or replacement attacks. We take advantage of FSTs' nondeterminism to capture all potential attack behaviours, because assuming a specific (e.g., deterministic) attack mapping may be too restrictive. As a result, FST models may be considered an overestimation of the actual attack activities. Thus, they implement the worst-case scenario.

One of the most fundamental attack strategies for this problem is the all-out attack strategy. This strategy encompasses any other attacks that utilize the same set of compromised events. In this model, the attacker can attack at any opportunity. The main focus of an intrusion detection problem is the attacker's strategy over a set of compromised events. Specifically, we design an effective intrusion detection module (IDM) based on a given attack strategy, such as a bounded or replacement strategy. The efficacy of the IDM is

directly tied to the type of attack strategy employed. Since an all-out attack encompasses all other techniques, an IDM that can effectively counter an all-out attack will also be effective against any other strategy.

Example 3. As depicted in Figure 6, finite-state transducers can encode different attack strategies. The (nondeterministic) deletion attack is shown in Figure 6a. When an event e fires, it allows the attacker to either delete $e \in \Sigma_{com}$ or leave it unchanged. The (nondeterministic) injection attack is represented in Figure 6b, indicating that a finite number of events from Σ_{com} can be inserted before or after received events. A replacement attack is illustrated by the FST in Figure 6c, where an event from Σ_{com} can be replaced by any event from Σ_{com} . Finally, the all-out attack is modelled by the FST in Figure 6d. This strategy includes the deletion, insertion, and replacement attack strategies over the same set of compromised events.

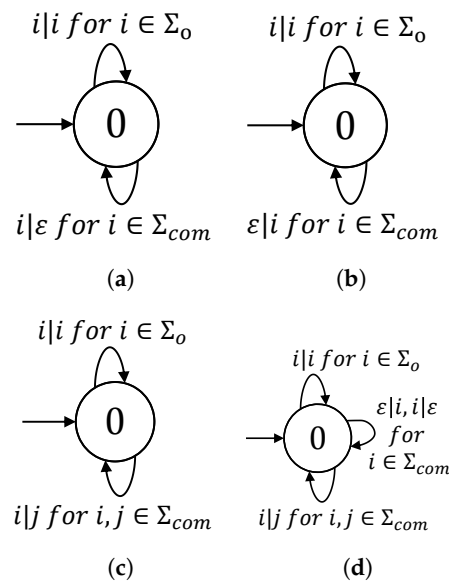


Figure 6. Attack modelling with finite state transducers. (a) Deletion attack A_{Del} ; (b) Insertion attack A_{Ins} ; (c) Replacement attack A_{Rep} ; (d) All-out attack A_{All} .

If no prior knowledge about the attack method is available, we presume that the adversary is using an all-out attack approach. The transducer that results from reversing the input and output alphabets and the input and output labels of each transition of T is referred to as the inverse of T , denoted by T^{-1} .

Remark 1. An FST is a strong modelling tool for attack manipulations. Particularly, it allows the inversion of an attack; here we have $A_{Del} = A_{Ins}^{-1}$ and the composition of multiple attacks which $A_{All} = A_{Del} \circ A_{Ins} \circ A_{Rep}$.

Corollary 1. $A_{Rep} = A_{Rep}^{-1}$, $A_{Ins} = A_{Del}^{-1}$, and $A_{Del} = A_{Ins}^{-1}$ hold.

Proposition 1. An attacker that uses the all-out attack strategy A_{All} can always reverse its attack action, i.e., $A_{All} = A_{All} \circ A_{All}^{-1}$.

Proof. We have $A_{All} = A_{Del} \circ A_{Ins} \circ A_{Rep}$. Based on Corollary 1, $A_{All}^{-1} = A_{Ins}^{-1} \circ A_{Del}^{-1} \circ A_{Rep}^{-1} = (A_{Del} \circ A_{Ins})^{-1} \circ A_{Rep}^{-1} = (A_{Del} \circ A_{Ins} \circ A_{Rep})^{-1}$. \square

The observable events sent by plant G are subject to nondeterministic revisions by the attacker A before they are received by the supervisor S , i.e., A may choose different attack actions for the same observation input.

Model of the Plant under Sensor Channel Attacks

Definition 3. Given two FSTs $T_1 = (Q_1, I_1, O_1, \eta_1, q_{10}, Q_{1m})$ and $T_2 = (Q_2, I_2, O_2, \eta_2, q_{20}, Q_{2m})$ with $O_1 = I_2$. The sequential composition of T_1 and T_2 , denoted by $T = T_1 \circ T_2$, is defined as $T = (Q_1 \times Q_2, I_1 \times O_2, \zeta, (q_{10}, q_{20}), Q_{1m} \times Q_{2m})$, where

$$\zeta((q_1, q_2), i, o) = \begin{cases} (q'_1, q'_2), & \text{if } \exists u \in O_1 : (q_1, i, u, q'_1) \in \eta_1 \wedge (q_2, u, o, q'_2) \in \eta_2, \\ (q'_1, q_2), & \text{if } (i, o) \in I_1 \times \{\varepsilon\} \wedge (q_1, i, o, q'_1) \in \eta_1, \\ (q_1, q'_2), & \text{if } (i, o) \in \{\varepsilon\} \times O_2 \wedge (q_2, i, o, q'_2) \in \eta_2, \\ \emptyset, & \text{otherwise.} \end{cases}$$

We remark that in order to compose two FSTs $T_1 \circ T_2$, the output alphabet of T_1 must match the input alphabet of T_2 . The sequential composition is obtained by merging the transitions where the output of T_1 is used as the input of T_2 . Transitions with ε output in T_1 (ε input in T_2) are retained since they can be triggered without affecting $T_2(T_1)$. Other transitions are discarded.

Example 4. Consider the networked control system of Example 1, where the sensor channel ch_{s1} is attacked, i.e., $\Sigma_{com} = \Sigma_{s1} = \{a, d\}$. The all-out attack strategy over the set of compromised events is encoded by FST A as shown in Figure 7a. Based on Definition 3, the serial composition of G and A is shown in Figure 7b. Notice that unobservable event e is associated with ε as the output in FST G . This is obvious since the projection function P_o masks it from the supervisor. FST $G \circ A$ captures the possible event changes made by A that are covert to the supervisor S , i.e., $L_{in}(G \circ A \circ S) \subseteq L(S)$.

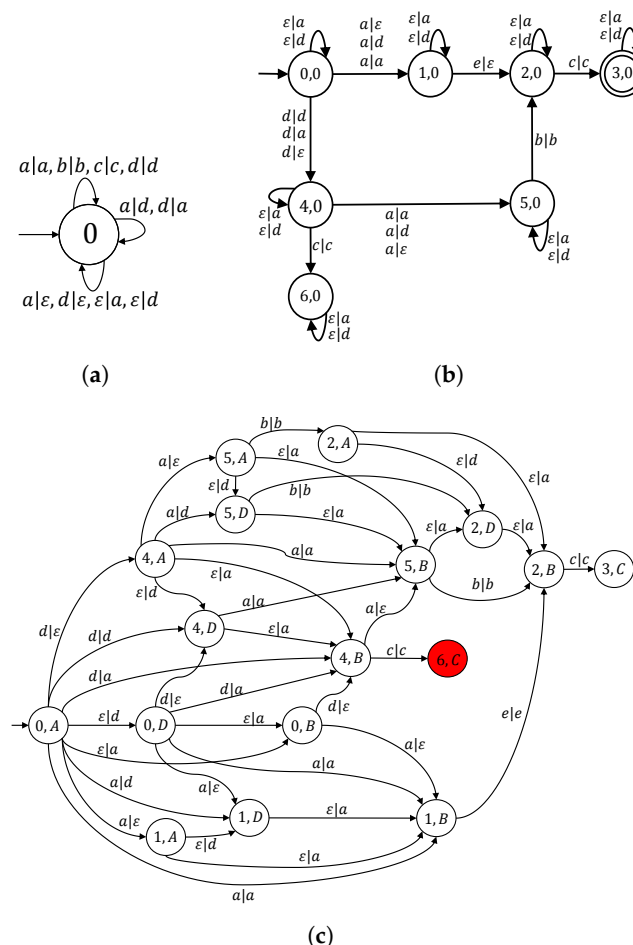


Figure 7. System under all-out attack. (a) All-out attack A . (b) Plant G under sensor attack: $G_A = G \circ A$. (c) Attacked closed-loop system: $S/G_A = G \circ A \circ S$.

5. Defence Mechanism

To guarantee safety, we employ a security mechanism known as an insertion function, which is commonly used to enforce opacity [38,39]. The insertion function, as depicted in Figure 8, acts as an interface between the system and external observers, where it receives the system's output behaviour, adds fictitious events when necessary, and then releases the modified output. The set of permissible events to be added is represented by Σ_o . We assume that the attacker is unable to distinguish between an added event and its authentic counterpart in Σ_o .

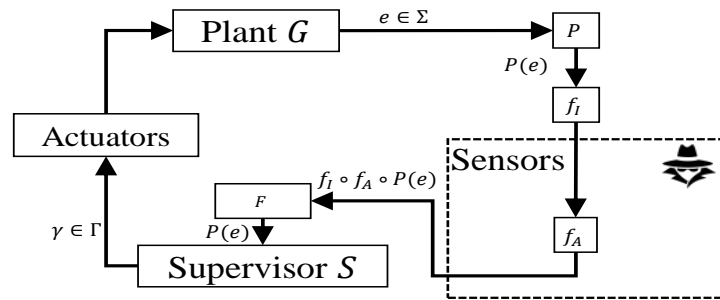


Figure 8. Controlled system under insertion function and supervisor filter against sensor deception attack.

5.1. Insertion Function

Given a finite state automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$, an insertion function is defined as a deterministic function $f_I : Q \times \Sigma_o \rightarrow \Sigma_o$ that outputs an inserted event based on the current state and the current observed event. Such an insertion function is represented by an input deterministic finite state transducer $I_T = (Q, \Sigma_o, \Sigma_o, f_I, q_0, Q_m)$, and is referred to as an insertion transducer. It has the same structure as G but is characterized with a different transition function.

Based on the work of Wu et al. [40], an insertion function has to be admissible. The admissibility property is a requirement for insertion functions, which mandates that insertion functions must be defined for all $P(L(G))$. This property is crucial since it ensures that the system's behaviour cannot be tampered with or hindered.

Definition 4. Given G , an insertion function is admissible if it inserts (possibly ϵ) on every observable event from G , i.e., $L_{in}(I_T) = P(L(G))$.

Accordingly, the admissibility property prevents the insertion function from blocking the controlled system after receiving an unexpected event from the plant. Safety property is an output property indicating that the output behaviour after insertion must include at least one safe event.

The projection over the set of safe events is recursively defined as follows: $P_{safe} : \Sigma^* \rightarrow \Sigma_o^*$

$$P_{safe}(\epsilon) = \epsilon \text{ and } P_{safe}(se) = \begin{cases} P_{safe}(s)e, & \text{if } e \in \Sigma_{safe} \\ P_{safe}(s), & \text{if } e \in \Sigma_{com} \end{cases}$$

Definition 5. Given G , a set of safe events Σ_{safe} and a set of compromised event Σ_{com} , an insertion function is safe if for each received compromised event the output word includes at least one safe event, that is, for each $q \in Q$ and $e \in \Sigma_{com}$, $P_{safe}[f_I(q, e)] \neq \epsilon$ holds.

Deterministic safe events insertion is an output property indicating that at a given state $q \in Q$, safe events are deterministically inserted.

Definition 6. Given G , set of safe events Σ_{safe} and a set of compromised event Σ_{com} , an insertion function is safe output deterministic if for each $q \in Q$ there do not exist $e, e' \in \Sigma_o$ such that $P_{safe}[f_I(q, e)] = P_{safe}[f_I(q, e')]$.

5.2. Supervisor Filter Design

In this subsection, we propose a new structure retrieved from the insertion transducer, called the supervisor filter, whose role is to recover the original language generated by the plant by handling the input events received by the insertion function and the attacker. The construction of such a module is defined in the following Algorithm 1.

Algorithm 1 Construction of the supervisor filter F

Input: A finite state transducer I_T .

Output: Supervisor filter F

Step 1. Compute inversion of I_T

$$F = (Q_f, I_f, O_f, \eta_f, q_{0f}) = I_T^{-1}$$

Step 2. Projection of the input language over Σ_{safe}

for each transition (q, u, v, q') **do**

$$(q, u, v, q') = (q, P_{safe}(u), v, q')$$

Step 3. Removing inputs possibly inserted by the attacker

for each $q \in Q_f$ and $e \in \Sigma_{com}$ **do**

if $\eta_f(q, e, \cdot)$ is not defined **then**

$$\eta_f(q, e, \epsilon) = q$$

return F

Example 5. Consider again the plant G shown in Figure 5a, where $\Sigma_o = \{a, b, c, d\}$ and $\Sigma_{com} = \{a, d\}$ are transmitted through sensor channel ch_{s1} . Its supervisor is depicted in Figure 5b, which guarantees that the damaging state $Q_{dmg} = \{6\}$ is unreachable. As shown in Figure 7c, the all-out attacker A_{all} can mislead the supervisor to generate a sequence of control patterns, which allows the plant to reach the damaging state (6, C). To prevent this, we propose the insertion transducer T_I encoded in Figure 9a and the associated filter F presented in Figure 9b.

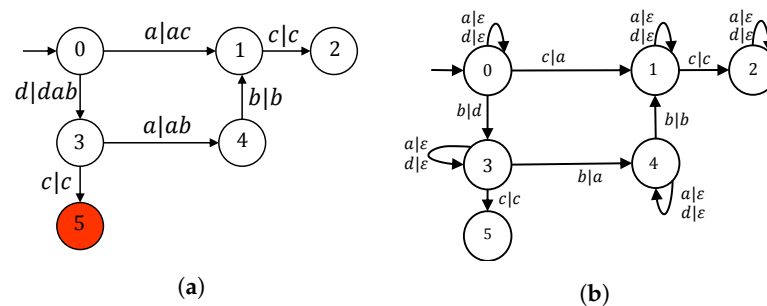


Figure 9. Defence mechanism. (a) Insertion transducer; (b) Supervisor filter.

Proposition 2. $L_{out}(G \circ I_T \circ A \circ F) = P(L(G))$.

Proof. It suffices to show that the language passing to the supervisor is exactly $P(L(G))$. By construction, we have $F = I_T^{-1}$, and $L_{out}(F) = L_{in}(I_T)$. Based on Definition 4, we have $L_{in}(I_T) = P(L(G))$, and then $L_{out}(F) = P(L(G))$. \square

With the presence of the filter and the insertion function, the sensor attack A does not influence the controllability. Specifically, the filter recovers all the original outputs of the plant, under which the supervisor achieves the control goal. This result differs from the majority of previous studies [3,18,23,36], since the insertion function and the integrated filter, which are modelled by FSTs instead of automata, have more power in safely countering the attacker actions by exploiting the safe part of the plant. The supervisor control commands depend on the filter's output language, which coincides with $L(G)$, i.e., $L_{in}(F) = L(G)$ instead of receiving corrupted observation from the attacker.

5.3. A Comparative Analysis of the Proposed Method with Existing Methods

In this subsection, we compare the proposed defence mechanism with state-of-the-art approaches based on the provided table, which considers the model type, attack types, strategy, methodologies, advantages, and disadvantages of each method (Table 1).

Table 1. Comparison of the defence mechanisms for discrete event systems against various types of attacks.

Reference	Model Type	Attack Types	Strategy	Methodologies	Advantages	Disadvantages
Our work	DES	Sensor	Secure control	Automaton, supervisory control theory	Being robust to many attacks	Exponential computational complexity
[21,41]	DES	Middleman	Detection	Automaton, supervisory control theory	Taking proper actions under attack	Unnecessary loss of resource
[23]	DES	Actuator, sensor	Detection	Automaton, supervisory control theory	Handling multiple integrity attack	Unnecessary loss of resources
[3,6]	DES	Sensor	Secure control	Automaton, supervisory control theory	Being robust to many attacks	Exponential computational complexity
[18]	DES	Sensor	Secure control	Supervisory control theory, game theory	Handling unknown attacks	Exponential computational complexity
[42]	DES	Sensor	Secure control	Petri nets, supervisory control theory	Compact mode	Exponential computational complexity
[43]	DES	Actuator, sensor	Secure control	Petri nets, supervisory, control theory	Compact model	Known attack structures

Comparing the proposed method with the existing ones, we observe that the proposed method provides robustness against sensor attacks and focuses on the secure control strategy. However, it has exponential computational complexity, which is a common disadvantage shared by several other methods, such as those in Refs. [3,6,18,42]. In contrast, the methods such as those in Refs. [21,23,41] disable all controllable events once an attack is detected, which may lead to unnecessary loss of resources. Furthermore, the method in Ref. [43] is limited by its reliance on known attack structures, potentially leaving the system vulnerable to novel or unknown attacks. This work, along with those in Refs. [3,6,18,42], focuses on secure control strategies to maintain system safety even under attack. This proactive approach helps us ensure the stability and integrity of the system, despite the exponential computational complexity.

6. Study Case: Non-Exposure of Vehicle Location

Let us consider a scenario where a vehicle travels on a public road, and the vehicle's location should be kept confidential, as shown in Figure 10. The vehicle is equipped with Global Positioning System (GPS) sensors that provide its location, which is used by the vehicle's control system to navigate safely. However, an attacker may try to compromise the sensors to extract the vehicle's location, which could be used for malicious purposes such as tracking or theft. The automaton G associated with our model is shown in Figure 11a, where $\Sigma = \Sigma_c = \Sigma_o = \{e, w, n, s, e^*, w^*, n^*, s^*\}$ and $\Sigma_{com} = \{e^*, w^*, n^*, s^*\}$ are transmitted through compromised GPS sensor channels. Its supervisor is depicted in Figure 11b, which guarantees that the damaging state $Q_{dmg} = \{q_2, q_{12}, q_{22}\}$ is unreachable. The all-out attacker A_{all} can mislead the supervisor to generate a sequence of control patterns, which allows the plant to reach the damaging state. To prevent this, we propose the insertion transducer T_I encoded in Figure 12a and the associated filter F presented in Figure 12b.

We can use the proposed defence mechanism based on an insertion function and a supervisory filter to protect the vehicle's location. The system's plant is modelled as a finite state automaton (FSA) $G = (Q, \Sigma, \delta, q_0, Q_m)$. The set of events Σ includes the GPS sensor readings, which provide the vehicle's directions.

The insertion function f_I is defined as a deterministic function that takes as input the current state and the current sensor reading and outputs a modified sensor reading. For example, we can define $f_I(q, x) = y$, where q is the current state, x is the original sensor reading, and y is the modified sensor reading. The insertion function can insert a fictitious

location that deviates from the actual location to confuse the attacker. The set of allowed events to be inserted is Σ_o , which includes all possible sensor readings.

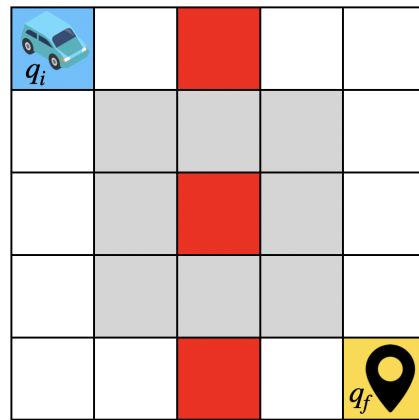


Figure 10. Vehicle grid: The vehicle starts in the blue cell denoted by q_i , the shaded area is considered hostile, and the sensor readings in this area are compromised.

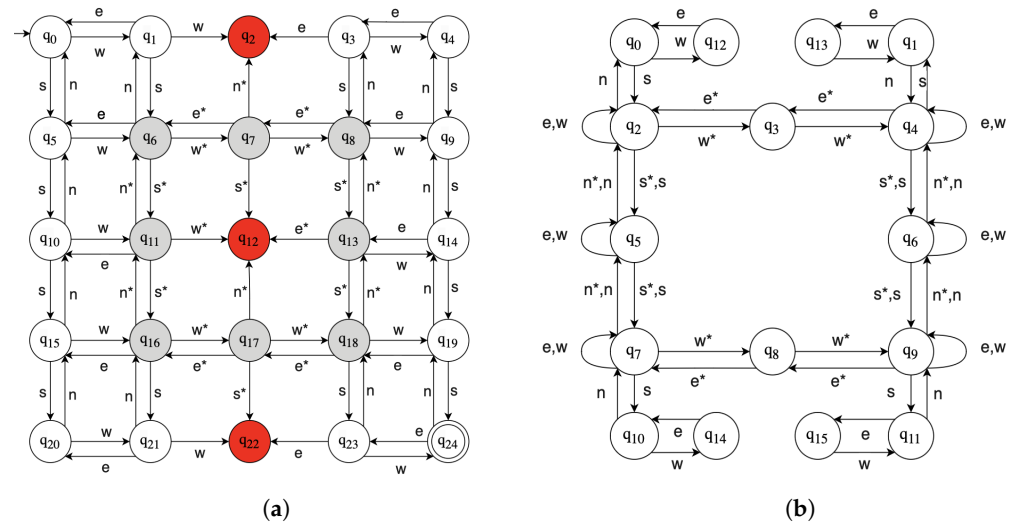


Figure 11. Closed loop system. (a) Model of the vehicle in the grid G : $\Sigma = \Sigma_c = \Sigma_o = \{e, w, n, s, e^*, w^*, n^*, s^*\}$. (b) Supervisor realisation.

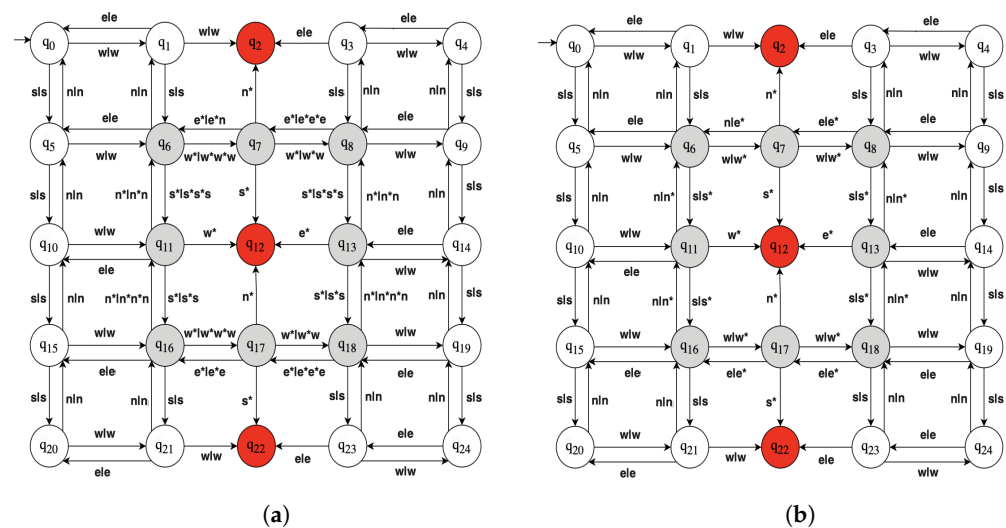


Figure 12. Defence mechanism. (a) Insertion transducer f_I . (b) Supervisor filter.

To ensure safety, we design the insertion function to be admissible, i.e., it should insert an event on every observable event from G . We also ensure that the insertion function is safe, which means that for each received compromised event (e.g., GPS sensor reading), the output word includes at least one safe event.

The supervisor filter F is constructed by taking the inverse of the insertion function I_T and removing the inputs that the attacker possibly inserts. This filter allows the supervisor to recover the original language generated by the plant, which is used to navigate and avoid damaging states.

7. Conclusions and Future Work

In this paper, we consider the problem of sensor deception attacks in a closed-loop control system, where the plant is abstracted as a discrete event system controlled by a supervisor through sensors and actuators. We propose a novel defence mechanism based on an insertion function that safeguard the system's compromised behaviour by exploiting the safe channels. The proposed approach offers two significant advantages. First, by utilizing nondeterministic finite-state transducers, we account for a worst-case attack scenario and effectively capture a broad range of attacks such as insertion, deletion, or replacement attacks. This approach ensures a robust defence against various types of attacks. Second, the proposed approach integrates a filter for post-attack language recovery while keeping the existing supervisor intact, rather than synthesizing a new supervisor. The insertion function and the filter cooperate to control the system and confuse the intruder without confusing the supervisor.

This work opens several avenues for future investigations. First, we are interested in extending the proposed method to the scope of DESs under actuator attacks. Second, we plan to synthesize an optimal insertion function that safeguards the system with the minimal use of safe channels. Finally, we intend to develop an algorithm for directly synthesizing the insertion function.

Author Contributions: Methodology, A.L. and I.S.; Software, M.A.E.-M.; Formal analysis, H.E.; Resources, M.S.; Supervision, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the Guangzhou Innovation and Entrepreneurship Leading Team Project Funding under Grant No. 202009020008, Researchers Supporting Program number (RSPD2023R704), King Saud University, Riyadh, Saudi Arabia, and the Science and Technology Fund, FDCT, Macau SAR, under Grant No. 0083/2021/A2.

Data Availability Statement: Not applicable.

Acknowledgments: The authors present their appreciation to King Saud University for funding this research through Researchers Supporting Program number (RSPD2023R704), King Saud University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tian, Z.; Jiang, X.; Liu, W.; Li, Z. Dynamic energy-efficient scheduling of multi-variety and small batch flexible job-shop: A case study for the aerospace industry. *Computers Ind. Eng.* **2023**, *178*, 109111. [\[CrossRef\]](#)
2. Fritz, R.; Zhang, P. Modeling and detection of cyber attacks on discrete event systems. *IFAC-PapersOnLine* **2018**, *51*, 285–290. [\[CrossRef\]](#)
3. Su, R. Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations. *Automatica* **2018**, *94*, 35–44. [\[CrossRef\]](#)
4. Wang, Y.; Pajic, M. Attack-resilient supervisory control with intermittently secure communication. In Proceedings of the IEEE 58th Conference on Decision and Control (CDC), Nice, France, 11–13 December 2019; pp. 2015–2020.
5. Lin, L.; Zhu, Y.; Su, R. Towards bounded synthesis of resilient supervisors. In Proceedings of the IEEE 58th Conference on Decision and Control (CDC), Nice, France, 11–13 December 2019; pp. 7659–7664.
6. Meira-Góes, R.; Marchand, H.; Lafortune, S. Towards resilient supervisors against sensor deception attacks. In Proceedings of the IEEE 58th Conference on Decision and Control (CDC), Nice, France, 11–13 December 2019; pp. 5144–5149.
7. Cong, X.; Fanti, M.P.; Mangini, A.M.; Li, Z. On-line verification of current-state opacity by Petri nets and integer linear programming. *Automatica* **2018**, *94*, 205–213. [\[CrossRef\]](#)

8. Dibaji, S.M.; Pirani, M.; Flamholz, D.B.; Annaswamy, A.M.; Johansson, K.H.; Chakraborty, A. A systems and control perspective of CPS security. *Annu. Rev. Control* **2019**, *47*, 394–411. [\[CrossRef\]](#)
9. Rashidinejad, A.; Wetzels, B.; Reniers, M.; Lin, L.; Zhu, Y.; Su, R. Supervisory control of discrete-event systems under attacks: An overview and outlook. In Proceedings of the 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 1732–1739.
10. Duo, W.; Zhou, M.; Abusorrah, A. A survey of cyber attacks on cyber physical systems: Recent advances and challenges. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 784–800. [\[CrossRef\]](#)
11. Zhang, Q.; Seatzu, C.; Li, Z.; Giua, A. Selection of a stealthy and harmful attack function in discrete event systems. *Sci. Rep.* **2022**, *12*, 16302. [\[CrossRef\]](#)
12. Zhang, H.; Feng, L.; Li, Z. A learning-based synthesis approach to the supremal nonblocking supervisor of discrete-event systems. *IEEE Trans. Autom. Control* **2018**, *63*, 3345–3360. [\[CrossRef\]](#)
13. Zhang, H.; Feng, L.; Wu, N.; Li, Z. Integration of learning-based testing and supervisory control for requirements conformance of black-box reactive systems. *IEEE Trans. Autom. Sci. Eng.* **2017**, *15*, 2–15. [\[CrossRef\]](#)
14. Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; Teneketzis, D. Diagnosability of discrete-event systems. *IEEE Trans. Autom. Control* **1995**, *40*, 1555–1575. [\[CrossRef\]](#)
15. Zhu, G.; Li, Z.; Wu, N. Model-based fault identification of discrete event systems using partially observed Petri nets. *Automatica* **2018**, *96*, 201–212. [\[CrossRef\]](#)
16. Zhu, G.; Li, Z.; Wu, N.; Al-Ahmari, A. Fault identification of discrete event systems modeled by Petri nets with unobservable transitions. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *49*, 333–345. [\[CrossRef\]](#)
17. Paoli, A.; Sartini, M.; Lafortune, S. Active fault tolerant control of discrete event systems using online diagnostics. *Automatica* **2011**, *47*, 639–649. [\[CrossRef\]](#)
18. Wakaiki, M.; Tabuada, P.; Hespanha, J.P. Supervisory control of discrete-event systems under attacks. *Dyn. Games Appl.* **2019**, *9*, 965–983. [\[CrossRef\]](#)
19. Gao, C.; Seatzu, C.; Li, Z.; Giua, A. Multiple attacks detection on discrete event systems. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 2352–2357.
20. Carvalho, L.K.; Wu, Y.C.; Kwong, R.; Lafortune, S. Detection and prevention of actuator enablement attacks in supervisory control systems. In Proceedings of the 13th International Workshop on Discrete Event Systems (WODES), Xi'an, China, 30 May–1 June 2016; pp. 298–305.
21. Lima, P.M.; Alves, M.V.; Carvalho, L.K.; Moreira, M.V. Security against network attacks in supervisory control systems. *IFAC-PapersOnLine* **2017**, *50*, 12333–12338. [\[CrossRef\]](#)
22. Lima, P.M.; Carvalho, L.K.; Moreira, M.V. Detectable and undetectable network attack security of cyber-physical systems. *IFAC-PapersOnLine* **2018**, *51*, 179–185. [\[CrossRef\]](#)
23. Carvalho, L.K.; Wu, Y.C.; Kwong, R.; Lafortune, S. Detection and mitigation of classes of attacks in supervisory control systems. *Automatica* **2018**, *97*, 121–133. [\[CrossRef\]](#)
24. Liu, Y.; Ning, P.; Reiter, M.K. False data injection attacks against state estimation in electric power grids. *ACM Trans. Inf. Syst. Secur. TISSEC* **2011**, *14*, 1–33. [\[CrossRef\]](#)
25. Lin, L.; Thuijsman, S.; Zhu, Y.; Ware, S.; Su, R.; Reniers, M. Synthesis of successful actuator attackers on supervisors. *arXiv* **2018**, arXiv:1807.06720.
26. Su, R. A cyber attack model with bounded sensor reading alterations. In Proceedings of the American Control Conference (ACC), Seattle, WA, USA, 24–26 May 2017; pp. 3200–3205.
27. Góes, R.M.; Kang, E.; Kwong, R.; Lafortune, S. Stealthy deception attacks for cyber-physical systems. In Proceedings of the 56th Annual Conference on Decision and Control (CDC), Melbourne, Australia, 12–15 December 2017; pp. 4224–4230.
28. Meira-Góes, R.; Kwong, R.; Lafortune, S. Synthesis of sensor deception attacks for systems modeled as probabilistic automata. In Proceedings of the American Control Conference (ACC), Philadelphia, PA, USA, 10–12 July 2019; pp. 5620–5626.
29. Meira-Góes, R.; Kwong, R.H.; Lafortune, S. Synthesis of optimal multi-objective attack strategies for controlled systems modeled by probabilistic automata. *IEEE Trans. Autom. Control* **2021**, *67*, 2873–2888. [\[CrossRef\]](#)
30. Meira-Góes, R.; Kang, E.; Kwong, R.H.; Lafortune, S. Synthesis of sensor deception attacks at the supervisory layer of cyber-physical systems. *Automatica* **2020**, *121*, 109172. [\[CrossRef\]](#)
31. Lin, L.; Thuijsman, S.; Zhu, Y.; Ware, S.; Su, R.; Reniers, M. Synthesis of supremal successful normal actuator attackers on normal supervisors. In Proceedings of the American Control Conference (ACC), Philadelphia, PA, USA, 10–12 July 2019; pp. 5614–5619.
32. Lin, L.; Zhu, Y.; Su, R. Synthesis of covert actuator attackers for free. *Discret. Event Dyn. Syst.* **2020**, *30*, 561–577. [\[CrossRef\]](#)
33. Zhu, Y.; Lin, L.; Su, R. Supervisor obfuscation against actuator enablement attack. In Proceedings of the 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 1760–1765.
34. Lin, L.; Su, R. Synthesis of covert actuator and sensor attackers. *Automatica* **2021**, *130*, 109714. [\[CrossRef\]](#)
35. Khoumsi, A. Sensor and actuator attacks of cyber-physical systems: A study based on supervisory control of discrete event systems. In Proceedings of the 8th International Conference on Systems and Control, Marrakesh, Morocco, 23–25 October 2019; pp. 176–182.
36. Su, R. On decidability of existence of nonblocking supervisors resilient to smart sensor attacks. *arXiv* **2020**, arXiv:2009.02626.
37. Ramadge, P.J.; Wonham, W.M. The control of discrete event systems. *Proc. IEEE* **1989**, *77*, 81–98. [\[CrossRef\]](#)

38. Saadaoui, I.; Li, Z.; Wu, N. Current-state opacity modelling and verification in partially observed Petri nets. *Automatica* **2020**, *116*, 108907. [\[CrossRef\]](#)
39. Labed, A.; Saadaoui, I.; Wu, N.; Yu, J.; Li, Z. Current-state opacity verification in discrete event systems using an observer net. *Sci. Rep.* **2022**, *12*, 21572. [\[CrossRef\]](#)
40. Wu, Y.C.; Lafortune, S. Synthesis of insertion functions for enforcement of opacity security properties. *Automatica* **2014**, *50*, 1336–1348. [\[CrossRef\]](#)
41. Lima, P.M.; Alves, M.V.S.; Carvalho, L.K.; Moreira, M.V. Security against communication network attacks of cyber-physical systems. *J. Control. Autom. Electr. Syst.* **2019**, *30*, 125–135. [\[CrossRef\]](#)
42. You, D.; Wang, S.; Seatzu, C. A liveness-enforcing supervisor tolerant to sensor-reading modification attacks. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *52*, 2398–2411. [\[CrossRef\]](#)
43. Wang, Y.; Li, Y.; Yu, Z.; Wu, N.; Li, Z. Supervisory control of discrete-event systems under external attacks. *Inf. Sci.* **2021**, *562*, 398–413. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.