Real-Time Attack-Recovery for Cyber-Physical Systems Using Linear Approximations

Lin Zhang[1], Xin Chen[2], Fanxin Kong[1], Alvaro A. Cardenas[3]
[1]*Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse NY*
[2]*Department of Computer Science, University of Dayton, Dayton OH*
[3]*Department of Computer Science and Engineering, University of California at Santa Cruz, Santa Cruz CA*
lzhan120@syr.edu, xchen4@udayton.edu, fkong03@syr.edu, alvaro.cardenas@ucsc.edu

2020 IEEE Real-Time Systems Symposium (RTSS) | 978-1-7281-8324-4/20/$31.00 ©2020 IEEE | DOI: 10.1109/RTSS49844.2020.00028

***Abstract*—Attack detection and recovery are fundamental elements for the operation of safe and resilient cyber-physical systems. Most of the literature focuses on attack-detection, while leaving attack-recovery as an open problem. In this paper, we propose novel attack-recovery control for securing cyber-physical systems. Our recovery control consists of new concepts required for a safe response to attacks, which includes the removal of poisoned data, the estimation of the current state, a prediction of the reachable states, and the online design of a new controller to recover the system. The synthesis of such recovery controllers for cyber-physical systems has barely investigated so far. To fill this void, we present a formal method-based approach to online compute a recovery control sequence that steers a system under an ongoing sensor attack from the current state to a target state such that no unsafe state is reachable on the way. The method solves a reach-avoid problem on a Linear Time-Invariant (LTI) model with the consideration of an error bound $\epsilon \geq 0$. The obtained recovery control is guaranteed to work on the original system if the behavioral difference between the LTI model and the system's plant dynamics is not larger than ϵ. Since a recovery control should be obtained and applied at the runtime of the system, in order to keep its computational time cost as low as possible, our approach firstly builds a linear programming restriction with the accordingly constrained safety and target specifications for the given reach-avoid problem, and then uses a linear programming solver to find a solution. To demonstrate the effectiveness of our method, we provide (a) the comparison to the previous work over 5 system models under 3 sensor attack scenarios: modification, delay, and reply; (b) a scalability analysis based on a scalable model to evaluate the performance of our method on large-scale systems.**

***Index Terms*—cyber-physical systems, security, sensor attacks, recovery, real-time**

I. Introduction

Cyber-Physical Systems (CPS) are engineered systems combining computation, communications, and physical resources. The integration of new technologies in the way we interact and control physical systems has enabled new applications such as autonomous vehicles, advanced manufacturing, and the smart grid [1]–[3]. However, these advances also raise new security issues that cannot be addressed with classical cyber security mechanisms designed for information technology systems [4]–[6].

One of the most pressing security risks in CPS is sensor spoofing attacks. When a controller acts on sensor information that is malicious, it can drive the physical system to unsafe states. In addition to software attacks, sensor spoofing can be done through transduction attacks, where the attacker manipulates a physical property that affects the reading of the sensor. For example, an attacker can exploit weaknesses in wheel speed sensors and affect antilock braking systems [7], or attack remotely a LiDAR sensor in vehicles to make them perceive non-existent objects [8]. Classical software security cannot prevent these attacks [9]–[13].

These new threats have motivated researchers to propose novel defense mechanisms against sensor attacks. Most of them fall into two categories: proactive resiliency and reactive resiliency. The first category assumes the control system might be under attack and attempts to minimize the impact from attacks [11], [14]–[17]. The second category focuses on detecting attacks by identifying anomalies between measurements and expected (estimated) values. Attack detection can be done using models of the system (i.e., system dynamics or trained models) [12], [13], [18]–[20] or via sensor correlation [21]–[24].

The vast majority of work to address this problem has focused on attack detection, but the key question that has not been addressed is how to respond to a detected attack. Even the surveys on attack detection emphasize how little work has been done in attack-recovery [18], [19]. In fact, a recent paper studied 32 CPS security surveys, and found that while most of them talk about attack-detection, only 8 of them described any form of response to detected attacks [25]. The authors conclude their paper by stating that "it is necessary that future work addresses the main benefit of intrusion detection: i.e., after detecting an attack, what should we do?"

In this paper we attempt to fill this void by proposing a formal-methods approach to compute a Piece-Wise Constant (PWC) control input sequence that recovers the CPS to a target state after a sensor attack is detected. Our work includes the following major contributions:

(i) A new attack-recovery architecture consisting of a *checkpointer* for keeping a finite set of historical states, *estimate reconstructor* for conservatively estimating the current system state, a *deadline calculator* for computing the length of the recovery control sequence, and a *recovery control calculator* to design the new control actions.
(ii) A formal method to conservatively estimate the current and future states with a control stepwise error bound $\epsilon \geq 0$ based on a Linear Time-Invariant (LTI) approximate

2576-3172/20/$31.00 ©2020 IEEE
DOI 10.1109/RTSS49844.2020.00028

model of the plant dynamics. The resulting estimations are guaranteed to contain the actual system states if the LTI model is an ϵ-approximation of the plant dynamics.

(iii) A way to formulate the reach-avoid problem the recovery controller has to design as a Linear Programming (LP) restriction. The solution set of the LP problem is contained in that of the reach-avoid problem, so a solution to the LP problem is also a solution to the reach-avoid problem, which is traditionally a hard problem to solve, even for LTI models.

The new recovery control is guaranteed to work on the original CPS if (a) it is applied immediately when the attack is detected, (b) we have a good estimate of when the attack started, and (c) the LTI model is an ϵ-approximation of the plant dynamics. On the other hand, our approach does not provide a guarantee on the success of finding a recovery control in any situation.

We argue that our proposed architecture contains the necessary blocks to design a safe recovery controller. In particular, (1) we need a *checkpointer* to keeps a window of the historical behavior of the system in order to remove false data for our attack-recovery computations. (2) When an attack is detected, we can no longer trust the information of the sensors, therefore we also develop a novel *estimate reconstructor*, which uses the historical data of the checkpointer to conservatively estimate the current system state. (3) If we know the current state or a conservative estimation of it, the next step is to find a feasible amount of time to safely steer the system to a target state. This is done by the *deadline calculator* which uses the method in contribution (ii) to find a time deadline. (4) Finally, the *recovery control calculator* uses the methods in contribution (iii) to compute a recovery control.

The rest of the paper is organized as follows. Section II discusses related work. Section III presents an overview of our recovery framework. Sections IV and V provide the details of each component in the framework. Section VI shows the experimental evaluation of our approach. The paper is concluded by Section VII.

II. Related Work

Before security was a concern, control systems had to deal with faults. *Fault Detection, Isolation, and Reconfiguration* (FDIR) [26] is an area of control where anomalies are detected using either a model-based detection system, or a purely data-driven system; this part of the process is also known as *Bad Data Detection*. Isolation is the process of identifying which device is the source of the anomaly, and reconfiguration is the process of recovering from the fault.

FDIR systems are good at detecting and eliminating faults caused by nature or accidents, but they do not provide security where faults are created by a strategic adversary. The main reason for this is that these protection systems generally assume independent, non-malicious failures, and in security, incorrect model assumptions are the easiest way for the adversary to bypass any protections [4]. For example, FDIR systems have been bypassed in the power grid [27] and chemical processes [28].

To fix the limitations of classical FDIR systems, there has been a lot of work in the literature on how to detect attacks on CPS. One of the areas that has been explored the most is how monitoring sensor (and actuation) values from physical observations, and control signals sent to actuators, can be used to detect attacks; this approach is usually called *physics-based* attack detection [18]. There are two classes of anomalies, historical anomalies [29], [30], and physical inconsistencies [31], [32].

Despite all this work on attack-detection, there has been relatively few attempts to respond to alerts from an intrusion detection system. Surveys on attack detection emphasize how little work has been done in attack-recovery [18], [19]. In fact, a recent paper studied 32 CPS security surveys, and found out that while most of them talk about attack-detection, only 8 of them described any form of response to detected attacks [25].

Although there are relatively few papers proposing attack-response methods, the most popular approach to respond to an attack is to ignore the spoofed sensors, obtain state estimates for the missing sensors, and then use the original controller to respond to the attack [9], [28], [33]. These papers however miss important concerns for recovering a CPS under attack: (1) using the original controller cannot guarantee that the system will remain safe under attacks, in fact these papers do not provide formal models to show why the recovery procedure will be safe, (2) for safety-critical systems, the *time* in which computations are performed is important in order to ensure the correctness of the system [34], so we need to include time deadlines for the recovery of the system, and (3) they assume that the defender knows the exact physical model of the system under attack; however this is rarely the case in practical systems.

In this paper, we address the above concerns by introducing (1) a formal method-based framework of recovery controller that is activated to steer the system to a target state when an attack is detected; (2) a deadline calculator that computes the number of control steps for the later recovery task; (3) a formalism that allows a pre-assumed largest behavioral difference between the actual system and our formal model which is an LTI system.

Although some of the reachability analysis techniques for dynamical systems have been applied to online monitoring the safety of CPS [35]–[38], there is no existing work to the best of our knowledge recovering a CPS under a sensor or actuator attack based on reachability. The work presented in the paper uses the efficient support function method [39], [40] to compute the reachable set overapproximations as well as a recovery deadline based on the LTI approximation of the CPS plant dynamics. Then an LP restriction is built for the reach-avoid problem describing the recovery task.

Although LTI models are one of the simplest categories of control systems, they are widely used in the analysis and synthesis of dynamical systems including mobile robots [41]–[43], medical devices [44], and aircraft systems [45], [46]

due to the efficient algorithms that can be applied to them. Some recent studies on the attack-resilient state estimation for CPS are also based on LTI models [47], [48]. Although most of the practical systems are nonlinear, the behavior of a stable nonlinear system in a short time period or a bounded neighborhood of an equilibrium can often be well approximated by an LTI system. This property gives birth to lots of linearization and hybridization techniques [49]–[51] that use linear models in the analysis of a nonlinear system. The correctness guarantee provided by our approach requires the existence of an LTI model which approximates the CPS plant with a pre-assumed accuracy. Such a model could be obtained as a linearization of the plant if its dynamics is nonlinear, or a data-driven model if the plant is given as a black box.

Our recovery framework can be seen as a variant of the simplex architecture [52] dealing with a "complex" controller and a "safety" controller.

III. Recovery of Cyber-Physical Systems

This section presents the system model, the threat model, and the design overview of our real-time recovery framework.

A. Scope of Our Work

As mentioned in the previous section, the work on attack detection for CPS is extensive, therefore in this paper we assume there is an attack detection system already in place, and our goal is to take the alerts that are generated by this intrusion detection system and respond to them in order to recover the physical system under control. We also assume that our intrusion detection system also gives us the time t_0 when the attack started; (this can be done by finding the time of a breakpoint of the time series of the residuals [18]).

B. Preliminaries

The CPS model considered in the paper is a physical process–which is also called a plant–controlled by a computer program or controller. The controller operates at every δ units of time, where $\delta > 0$ is called a *control step*. At the beginning of every control step, the controller reads the state estimate of the plant (represented by a set of real-valued variables $\{x_1, \ldots, x_n\}$), and using a control algorithm, the controller computes the control signals $\{u_1, \ldots, u_m\}$ which are sent to the actuators. For brevity, we collectively represent the variables $x_1(t), \ldots, x_n(t)$ by $\vec{x}(t)$ and $u_1(t), \ldots, u_m(t)$ by $\vec{u}(t)$, at time t.

The variable(s) $\vec{y}(t)$ denotes the sensor measurement received at the controller (which may be compromised) while $\vec{y}\,'(t)$ is the real (true) value of the plant. $\vec{x}(t)$ denotes the state estimate while $\vec{x}'(t)$ is the real state of the plant. $\vec{u}(t)$ denotes the control input computed by the controller while $\vec{u}'(t)$ is the real input to the plant. We assume that all state variables are visible to sensors, and therefore we have that $\vec{x} = \vec{y}$ and $\vec{x}' = \vec{y}'$ in the rest of the paper.

Example III.1. *DC motors are extensively used as actuators in electric vehicles and prototype autonomous cars. A DC motor is equipped with torque converter, transmission, shaft, and wheels, and provides rotary motion. The following ODE models the behavior of a DC motor*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K_T}{J} \\ -\frac{K_e}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u$$

such that x_1 denotes the angular velocity of the motor and x_2 denotes the armature current. The control input is denoted by u which is the voltage applied to the motor, it is updated every 0.02 seconds based on the current value of x_1 by a PID control scheme whose goal is to maintain the angular velocity along a specific (reference) value which could be different in different time periods. Fig. 1 illustrates an execution of the DC motor from the initial state $x_1(0) = 0$, $x_2(0) = 0$, where the parameters are set as $R = 1$, $L = 0.5$, $K_T = 0.01$, $b = 0.1$, $J = 0.01$, and $K_e = 0.01$. The blue dashed line in the figure shows the reference values for x_1.

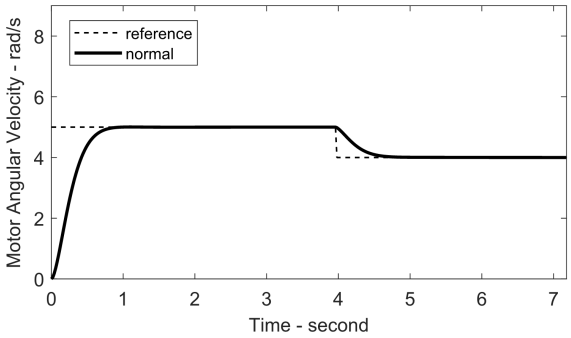


Fig. 1. An Execution of the CPS example

Threat Model. In this paper, our threat model consists of a CPS and an attacker who is able to modify the data sent from the sensor to the controller. In such a sensor attack, the data read from the sensor might not represent the actual plant or system state, i.e., $\vec{x}(t) \neq \vec{x}'(t)$ in Fig. 3, and the controller may produce an inappropriate control input based on the wrong state, so that a well-defined stability controller might become unstable, or a safety controller might steer the system to an unsafe state. Therefore, sensor attacks could lead to safety problems, and our goal in the paper is to present a framework to ensure the safety of CPS under sensor attacks.

The attack scenarios under our consideration are given as follows: (i) *Modification attack.* This attack modifies sensor data before it reaches the controller by adding or subtracting some values. Our technique does not restrict the number of dimensions can be compromised. For example, the value $\vec{x}(t)$ can be set to be $\vec{x}'(t) + (100, \ldots, 100)^T$, all dimensions compromised, or to $\vec{x}'(t) + (0, \ldots, 0, 100, \ldots, 100)^T$, partial dimensions compromised, starting from the beginning of the attack. (ii) *Replay attack.* The attacker sends the data from a previous time period to the controller. That is, $\vec{x}(t) = \vec{x}'(t-s)$ starting from the attack for some $s > 0$. (iii) *Delay attack.* The attack intentionally delays the data sent to the controller, i.e., $\vec{x}(t) = \vec{x}'(t_0)$ for a time period of d where t_0 is the start

time of the attack, and then $\vec{x}(t) = \vec{x}'(t-d)$ for $t \geq t_0 + d$. The Denial-of-Service (DoS) attack is also included in this category as the case that the delay is infinite.

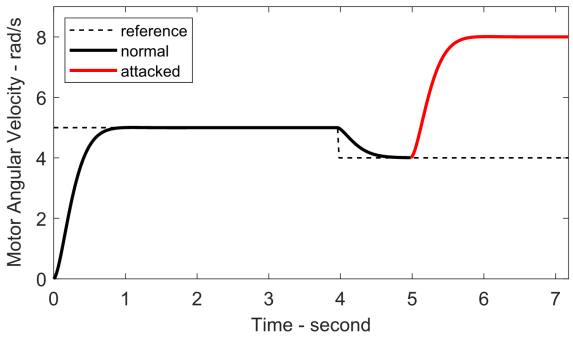


Fig. 2. The CPS example under a modification attack

Example III.2. *We show an example of modification attack in Fig.2. Starting from the time $t = 5$, the sensor data sent to the controller is modified by an attacker that adds a bias of 5 rad/s to x_1. Then after a small time period, the PID controller cannot maintain the motor angular velocity near the reference value which is 4 rad/s.*

C. Overview of the Real-Time Recovery Framework

Our proposed architecture for real-time recovery is presented in Fig. 3. Note that this architecture can be applied as an extension to an existing CPS and neither the original plant dynamics nor the control algorithm needs to be modified. The framework consists of four components, shown by the shaded boxes: (a) *recovery controller*, (b) *estimate reconstructor*, (c) *deadline calculator*, and (d) *checkpointer*. The following briefly describes the framework, we will provide details in Sections IV and V.

We consider two operating modes of the system: **normal** and **recovery** mode. As shown in Fig. 3, the attack detector determines whether the system is under attack (as mentioned at the beginning of this section, this part is outside the scope of our paper). The detector can utilize existing attack-detection mechanisms such as [12], [13], [18], [19]. Designing attack-detection methods is one of the most prolific areas in CPS security, but as noted in the introduction, most attack-detection papers do not propose what to do after an alarm is raised. This gap is the focus of this paper; in particular we propose that when the detector discovers attacks, the system will be switched to the recovery mode; otherwise, the system operates in normal mode.

Recovery Mode. In the recovery mode, the *recovery controller* will take over control of the system. This controller computes a Piece-Wise Constant (PWC) control sequence that is guaranteed to steer the system back to target state set before a safety deadline. This computation uses the reconstructed state estimate and safety deadline produced by the following two components. The *estimate reconstructor* rebuilds the state estimate at the time point when an attack is detected, which is

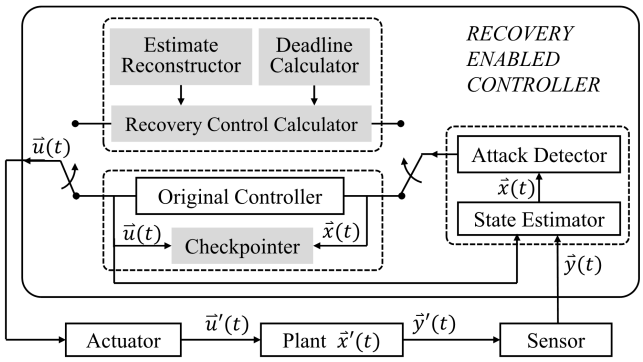


Fig. 3. Design Overview of real-time recovery. Recovery controller: Section IV. Checkpointer: Section V-A. Estimate reconstructor: Section V-B. Deadline calculator: Section V-C.

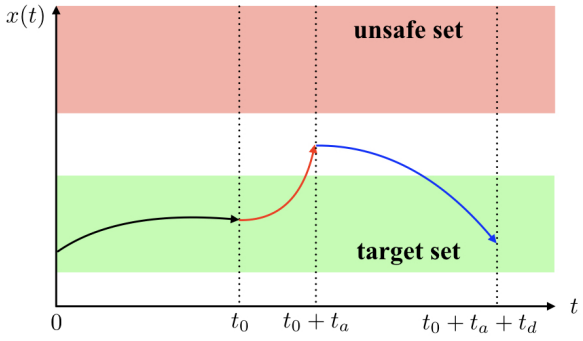


Fig. 4. Illustration of real-time recovery in the timeline.

also the start point the PWC control sequence starts from. This reconstruction uses trusted data given by the checkpointer. The *deadline calculator* calculates a safety deadline, which is a conservative estimation for the latest time when the system should be steered back to a target state set or the earliest time when the system may reach the unsafe state set. Serious consequences may occur if the system cannot be recovered by the safety deadline.

Normal Mode. In the normal mode, the original controller is used to control the system. Our framework does not modify this controller and uses the *checkpointer* to record historical data, i.e., state estimates $\vec{x}(t)$ and control inputs $\vec{u}(t)$. The recorded data will be used by the estimate reconstructor to rebuild the state estimate as mentioned above in the recovery mode. The checkpointer guarantees that the recorded data are kept for a window of time to accommodate the detection delay. The data within the window may be already compromised and thus temporarily treated as untrustworthy, while the data that is outside the window are treated as trustworthy.

An illustration of the real-time recovery framework in the timeline is shown in Fig. 4. The variable $x(t)$ denotes the state value of a safety-critical system state variable at the time t. Any state in the *unsafe set* should not be reachable, they are used to indicate catastrophic events such as crashing to the ground when the system is a UAV, or the collision of two vehicles when the system is a platoon of vehicles. The *target*

set is a neighbourhood of the reference states of the original system controller. The complement of the unsafe set is the *safe set*, however a system is only guaranteed to be currently safe when it is in a safe state.

Consider Fig. 4, where a sensor attack starts at t_0. First, before t_0, the system runs normally, i.e., uses the original controller and checkpoints historical data regularly. The system tracks the reference as shown by the black curve. Second, the attack detector raises an alert, but not before a certain detection delay, i.e., the time interval between the onset of an attack and detection of it, [9], [13], [18]. Thus, the attack is discovered at time $t_0 + t_a$, where t_a denotes the detection delay. During the delay, the system is affected by the attack so that it deviates from the reference state and further drifts out of the target set, as shown by the red curve. Third, after the attack is detected, the system is switched to the recovery mode at time $t_0 + t_a$. The recovery controller drives the system back to the target state set before $t_0 + t_a + t_d$, where t_d is the safety deadline, as shown by the blue curve. This controller also ensures that no unsafe state is reached during the recovery process.

IV. Real-Time Recovery Using PWC Control

In this section, we introduce an approach to compute a PWC control sequence for recovering in real-time a CPS under a sensor attack based on a Linear Time-Invariant (LTI) approximation of its plant dynamics. We show that the problem of finding such a sequence can be encoded by Linear Programming (LP), and the result is guaranteed to recover the original system when a conservative safe set and target set is considered on the LTI model.

Assume that the plant dynamics of the CPS are defined by a function $\vec{x}'(t) = \varphi(\vec{x}(0)', t, \vec{u}'(t))$, which is not necessarily known in our approach, such that $\vec{x}'(t)$ denotes the system state at the time t, and $\vec{u}'(t)$ is the control function. As we said, we only consider the discrete states $\vec{x}'(0), \vec{x}'(\delta), \vec{x}'(2\delta), \ldots$ where $\delta > 0$ is the control stepsize, and the control input in each step is constant. Then an *LTI approximation* for the plant dynamics is of the form

$$\vec{x}_{k+1} = A\,\vec{x}_k + B\,\vec{u}_{k+1} \tag{1}$$

where $\vec{x}_k$ is an approximation of $\vec{x}'(k\delta)$, and $\vec{u}_{k+1} = \vec{u}'(t)$ such that $t \in [k\delta, (k+1)\delta]$. In other words, the LTI system (1) simulates the original system discretely by a stepsize δ using the same control inputs. We further call the LTI approximation an ϵ-approximation, if its state difference from the original dynamics by one step is bounded by some $\epsilon > 0$, i.e., for any state $\vec{s}$ in the system state space $\mathcal{X}$, any control input $\vec{c}$ in the control input range $\mathcal{U}$, we have that

$$\varphi(\vec{s}, \delta, \vec{c}) \;\in\; \{A\vec{s} + B\vec{c}\} \oplus \mathcal{B}_\epsilon \tag{2}$$

where $\mathcal{B}_\epsilon$ is the origin-centered ball whose radius is ϵ, and $\oplus$ denotes the Minkowski sum, i.e., $X \oplus Y = \{x + y \,|\, x \in X, y \in Y\}$ for any sets X, Y. The recovery control is obtained based on an ϵ-approximation of the original dynamics in our approach.

Why aren't we using an open-loop control only based on the LTI approximation? Even an ϵ-LTI approximation cannot ensure the approximation quality for a long-term run. Lemma IV.1 indicates that the overall approximation error accumulates along the number of steps and linear in ϵ. Although we may limit the overall error bound by reducing ϵ, the quality of a long-term run is still hard to improve due to the size of $|A|$. Therefore, the LTI model is only considered in the recovery work which is generally over a short time period.

Lemma IV.1. *Given an initial state $\vec{x}_0$ and a k-step control sequence $\vec{u}_1, \ldots, \vec{u}_k$, we use $\vec{x}'_i$ to denote the reachable state of the original system at $t = i\delta$ for $i = 1, \ldots, k$, and $\vec{x}_i$ to denote the reachable state of the LTI approximation after the i-th step. Then we have that $|\vec{x}'_i - \vec{x}_i| \le \epsilon \cdot \sum_{j=0}^{i-1} |A|^j$ where $|\cdot|$ denotes the maximum norm.*

Proof. Since $\vec{x}_i = A\vec{x}_{i-1} + B\vec{u}_i$ and $\vec{x}'_i \in \{\vec{x}_i\} \oplus \mathcal{B}_\epsilon$, we have that

$$\begin{aligned}
\vec{x}'_i \in& \{A\vec{x}'_{i-1} + B\vec{u}_i\} \oplus \mathcal{B}_\epsilon \\
\subseteq& A(\{A\vec{x}'_{i-2} + B\vec{u}_{i-1}\} \oplus \mathcal{B}_\epsilon) \oplus \{B\vec{u}_i\} \oplus \mathcal{B}_\epsilon \\
=& \{A^2\vec{x}'_{i-2} + AB\vec{u}_{i-1} + B\vec{u}_i\} \oplus A\mathcal{B}_\epsilon \oplus \mathcal{B}_\epsilon \\
\subseteq& \{A^k\vec{x}_0 + \sum_{j=0}^{i-1} A^j B\vec{u}_{i-j}\} \oplus \bigoplus_{j=0}^{i-1} A^j\mathcal{B}_\epsilon = \{\vec{x}_i\} \oplus \bigoplus_{j=0}^{i-1} A^j\mathcal{B}_\epsilon
\end{aligned}$$

Hence, the difference between $\vec{x}'_i$ and $\vec{x}_i$ is bounded by $\epsilon \cdot \sum_{j=0}^{i-1} |A|^j$. □

To *recover a CPS* we need to find a sequence of control inputs $\vec{u}_1, \ldots, \vec{u}_N$ which steers the system from the current state, to a state in the given target set when the sensor attack is detected. We say that the recovery is in *real-time* when the problem further requires that N should be no greater than a given deadline D.

There are three main challenges to solve the real-time recovery problem for a CPS. (i) The exact state of the system at the current time is unknown due to the sensor attack. (ii) The shortest length N for the recovery control sequence is unknown. Moreover, the existence of an N-step recovery control does not imply the existence an $(N+1)$-step recovery control, which means that we have to check the feasibility from the case $N = 1$. (iii) The computation time for obtaining the recovery control should be nearly same as the response time of the system controller, since we assume that the result will be applied immediately at the current step. We present an approach to address the challenges (ii), (iii), and leave the technique of estimating the current system state to the next section.

We seek to focus only on a fixed length for finding a recovery control in order to avoid checking all possible lengths which is computationally expensive. To do so, we require that the target set only contains *maintainable state* of the LTI system, and the recovery control problem on LTI systems asks to find a control sequence below a specific length steering the system from a conservative estimation of the current state to a maintainable target set while no unsafe state is reached.

Definition IV.1. *A state $\vec{s}$ of a given LTI system in the form of (1) is* maintainable*, if and only if there exists a control input $\vec{c}$ such that $(I - A)\vec{s} = B\vec{c}$, i.e., $\vec{s} = A\vec{s} + B\vec{c}$.*

Definition IV.2 (Recovery problem on LTI systems)**.** *Given a start state $\vec{x}_0$, a safe set X_S, a maintainable target set $X_T \subseteq X_S$, a LTI system in the form of (1), and a deadline D, the recovery problem asks whether there exists a control sequence $\vec{u}_1, \ldots, \vec{u}_N$ where $N \leq D$ steering the system from $\vec{x}_0$ to a state in X_T while all reachable states on the way are in X_S.*

Lemma IV.2. *If there is a solution for the recovery problem with $N < D$, then there is also a solution with $N = D$.*

Proof. Assume that $\vec{u}_1, \ldots, \vec{u}_N$ is a solution for the recovery problem such that $N < D$. Then the reachable state $\vec{x}_N$ from $\vec{x}_0$ after applying the N control inputs must be in the maintainable set X_T, and therefore there exists a control input $\vec{u}^*$ which maintains the system in the state $\vec{x}_N$ after one step. Hence, the solution $\vec{u}_1, \ldots, \vec{u}_N$ can be extended by concatenating $D-N$ many $\vec{u}^*$ control inputs, and the extended sequence is a solution whose length is D of the recovery problem. □

Finding the control inputs. Lemma IV.2 allows us to only focus on the case $N = D$ in looking for a solution for the recovery problem. We use the variable $\vec{x}_i$ to denote the state of the LTI system at the i-th step, $\vec{u}_i$ to denote the control input used in the i-th step, then the recovery run can be described by the constraint

$$\phi = (\vec{x}_D \in X_T) \wedge \bigwedge_{i=0}^{D} (\vec{x}_i \in X_s) \wedge \bigwedge_{i=0}^{D-1} (\vec{x}_{i+1} = A\vec{x}_i + B\vec{u}_{i+1})$$

and when X_T and X_S are defined by linear constraints, the recovery problem can be solved by finding a solution for the following LP problem:

$$\text{Find } \vec{x}_1, \ldots, \vec{x}_D \in \mathcal{X},\ \vec{u}_1, \ldots, \vec{u}_D \in \mathcal{U} \text{ s.t. } \phi \qquad (3)$$

where $\mathcal{X}$ is the state space and $\mathcal{U}$ is the range of all possible control inputs of the original system.

We use $\mathcal{E} = \epsilon \cdot \sum_{j=0}^{D-1} |A|^j$ to denote the maximum bound of the difference between the original system and its LTI approximation in D steps, and $\mathcal{B}_\mathcal{E}$ to denote the original-centered ball of radius $\mathcal{E}$, then the following theorem tells us that if we are able to find a recovery control sequence on the LTI system on more restrictive safe set and target set, then the result is also a recovery control sequence for the original system.

Theorem IV.1. *Assume that X'_S is the safe set and X'_T is the target set which is not necessarily maintainable of the original system. If $\vec{u}_1, \ldots, \vec{u}_D$ is obtained from a solution of the LP problem (3) such that $X_S \oplus \mathcal{B}_\mathcal{E} \subseteq X'_S$, $X_T \oplus \mathcal{B}_\mathcal{E} \subseteq X'_T$, then the control sequence can also used to recover the original system.*

Proof. By Lemma IV.1, for $i = 0, \ldots, D$, the difference between the original system state $\vec{x}'_i$ at the time $t = i\delta$ and the state $\vec{x}_i$ of its LTI approximation after the i-th step is bounded by $\mathcal{E}$, since $\vec{x}_0, \ldots, \vec{x}_D \in X_S$ and $\vec{x}_D \in X_T$, we have that $\vec{x}'_0, \ldots, \vec{x}'_D \in X_S \oplus \mathcal{B}_\mathcal{E} \subseteq X'_S$ and $\vec{x}'_D \in X_T \oplus \mathcal{B}_\mathcal{E} \subseteq X'_T$, i.e., the control sequence also recovers the original system. □

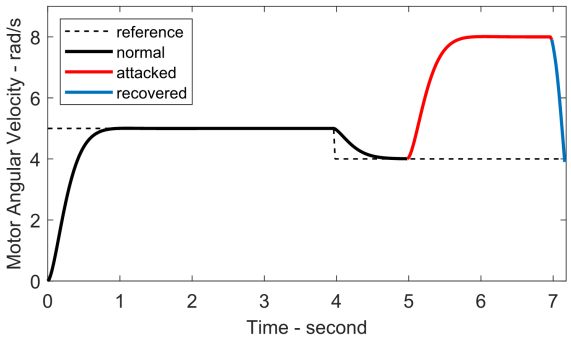


Fig. 5. Illustration of the system recovering from the sensor attack as shown in Example III.2

Example IV.1. *We show the effectiveness of our real-time recovery approach on the sensor attacks given in Example III.2. Since the original system is linear, our LTI approximation is directly obtained by evaluating the solution form of the ODE, and therefore the approximation error is just the Taylor approximation error of the matrix exponential. We more conservatively shrink the safe set from $0 \leq x_1 \leq 10$ to $1 \leq x_1 \leq 9$, and the target set from $3.5 \leq x_1 \leq 4.5$ to $3.7 \leq x_1 \leq 4.3$. Our approach successfully recovers the system to the target set while no state on the way is unsafe, as shown in Fig. 5.*

Extension to a set of start states. Since $\vec{x}_0$ is only an estimation of the system state when a sensor attack is detected, we should take into account that there may be an estimation error. To do so, we consider a set X_0 of start states which is a ball centered at $\vec{x}_0$ and its radius is $\xi > 0$ which is the assumption of the upper bound of the estimation error, i.e., $X_0 = \{\vec{x}_0\} \oplus \mathcal{I}_\xi$. The following theorem tells us that if a recovery control sequence is found only for $\vec{x}_0$ but under more restrictive safe set and target set, then it also works for all start state in X_0.

Theorem IV.2. *Given that $X_0 = \{\vec{x}_0\} \oplus \mathcal{I}_\xi$ is a conservative estimation of the start state, X_S is the safe set, X_T is the target set which is maintainable, and D is the deadline, then a solution of the following feasibility problem is a recovery control which works on all states in X_0.*

$$\text{Find } \vec{u}_1, \ldots, \vec{u}_D \in \mathcal{U} \text{ s.t. } \bigwedge_{i=0}^{D-1} (\vec{x}_{i+1} = A\vec{x}_i + B\vec{u}_{i+1})$$
$$\bigwedge_{i=0}^{D} \vec{x}_i \in (X_S \ominus A^i\mathcal{I}_\xi) \ \wedge\ \vec{x}_D \in (X_T \ominus A^D\mathcal{I}_\xi)$$

where $\ominus$ denotes the Minkowski difference, i.e., $X \ominus Y = \bigcap_{y \in Y} \{x - y \,|\, x \in X\}$.

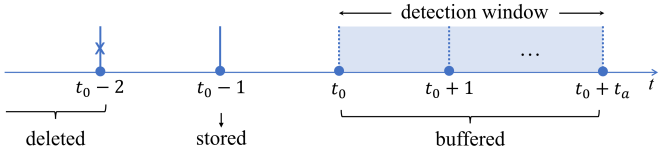


Fig. 6. The checkpointer uses the sliding window based checkpointing protocol proposed by [9].

Proof. We show that all reachable states from X_0 in D steps are contained in the safe set X_S and the final reachable set is contained in the target set X_T. For $i = 0, \ldots, D$, $\vec{x}_i = A^i\vec{x}_0 + \sum_{j=0}^{i-1} A^j B\vec{u}_{i-j}$. When $\vec{x}_0$ is replaced by the set X_0, then the reachable set at the i-th step becomes

$$X_i = A^i X_0 + \sum_{j=0}^{i-1} A^j B\vec{u}_{i-j} = A^i(\{\vec{x}_0\} \oplus \mathcal{I}_\xi) \oplus \{\sum_{j=0}^{i-1} A^j B\vec{u}_{i-j}\}$$
$$= \{A^i\vec{x}_0 + \sum_{j=0}^{i-1} A^j B\vec{u}_{i-j}\} \oplus A^i\mathcal{I}_\xi = \{\vec{x}_i\} \oplus A^i\mathcal{I}_\xi$$

In order to ensure $X_i \subseteq X_S$, we only need to require that $\vec{x}_i \in X_S \ominus A^i\mathcal{I}_\xi$ holds. The proof of the correctness of using the constraint $\vec{x}_D \in (X_T \ominus A^D\mathcal{I}_\xi)$ is similar. □

For simplicity, we may consider using the polyhedral or even box underapproximations for the set $X_S \ominus A^i\mathcal{I}\xi$, $X_T \ominus A^D\mathcal{I}_\xi$, and the problem in Theorem IV.2 becomes an LP problem. In the next section, we introduce our approach to construct a conservative estimation of the system state when a sensor attack is detected.

V. Supporting Components for Real-Time Recovery

In this section, we give a detailed description for the design of other components in our real-time recovery framework. We firstly show that we may use the checkpointing protocol method presented in [9] to obtain the nearest trustworthy sensor data, based on which a conservative estimation of the start state of recovery can be obtained using a reachability computation technique, and then a conservative deadline can be computed using a safety verification method.

A. Checkpointing Protocol

Historical state estimates during the detection delay are not trustworthy as they may be already compromised due to the attack. Thus, using them can result in unsuccessful recovery. To accommodate this, we apply the sliding window based checkpointing protocol proposed by [9] in this paper. The protocol provides trustful historical data (i.e., state estimates and control inputs) that can be used for reconstructing the state estimate at the time point when an attack is detected. We now summarize this protocol.

The protocol uses a detection window to capture the detection delay, where the length of the window equals the delay. The detection window slides forward as the time ticks. The protocol checkpoints the system, i.e., the state estimate $\vec{x}(t)$ and control input $\vec{u}(t)$, at every control step t. There are three steps: buffer, store, and delete.

i) Buffer. Fig. 6 shows an illustrative example. The detection window has a length of t_a. State estimates within the window, i.e., $\{\vec{x}(t_0), \ldots, \vec{x}(t_0 + t_a)\}$ in the time interval of $[t_0, t_0 + t_a]$, are first buffered, because they may be already corrupted and whether they are correct is still in question. Note that the recovery controller starts from the time step when an attack is detected. Thus, these state estimates cannot be used for reconstructing the state estimate $\vec{x}(t_0 + t_a)$ if we detect an attack at time $t_0 + t_a$.

ii) Store. State estimates that moved outside the detection window are considered to be trustful. They are stored and can be used for reconstructing the state estimate. As shown in Fig. 6, estimate $\vec{x}(t_0 - 1)$ lays outside the window, and thus $\{\vec{x}(t_0 - 1), \vec{u}(t_0 - 1)\}$ is stored and trusted to rebuild estimate $\vec{x}(t_0 + t_a)$ if detecting an attack at the time.

iii) Delete. The stored estimates and control inputs are discarded if they are no longer needed. $\{\vec{x}(t_0 - 2), \vec{u}(t_0 - 2)\}$ is discarded in the example.

B. State Estimate Reconstruction

We show that a conservative estimation X_0 for the starting state for recovery can be obtained using a reachability computation technique on an LTI ϵ-approximation of the CPS.

Assume that a sensor attack is detected t_a seconds after the attack begins at the time t_0, and there are in total $N_a = t_a/\delta$ control steps in between. Then, the sensor data at time t_0 is stored by the checkpointing protocol and it is the nearest trustworthy data based on which we will compute a conservative estimation for the actual system state at the time $t_0 + t_a$. We assume that this trustworthy state is $\vec{x}_a$.

According to the property of ϵ-approximation, we have that the actual system state at the time $t_0 + t_a$ must be contained in the set

$$\left\{A^{N_a}\vec{x}_a + \sum_{j=0}^{N_a-1} A^j B\vec{c}_{N_a-j}\right\} \oplus \bigoplus_{j=0}^{N_a-1} A^j\mathcal{B}_\epsilon \qquad (4)$$

where $\vec{c}_1, \ldots, \vec{c}_{N_a}$ is the control inputs used in the past N_a steps. Since the computation of the above set only involves linear operations such as linear transformation and Minkowski sum, a box overapproximation X_0 of it can be obtained from using the support function method [53], that is, the upper and lower bounds of the above set in the i-th dimension can be evaluated as

$$\vec{l_i}^T\left(A^{N_a}\vec{x}_a + \sum_{j=0}^{N_a-1} A^j B\vec{c}_{N_a-j}\right) + \sum_{j=0}^{N_a-1}\sqrt{\vec{l_i}^T A^j (A^j)^T \vec{l_i}}\epsilon,$$
$$\vec{l_i}^T\left(A^{N_a}\vec{x}_a + \sum_{j=0}^{N_a-1} A^j B\vec{c}_{N_a-j}\right) - \sum_{j=0}^{N_a-1}\sqrt{\vec{l_i}^T A^j (A^j)^T \vec{l_i}}\epsilon$$

respectively, such that $\vec{l_i}$ is the column vector whose i-th component is 1 and the others are 0. We may use this box

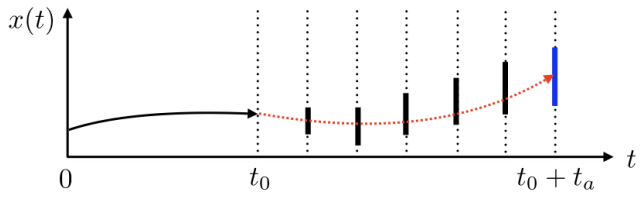


Fig. 7. Start state estimation. Line segments: overapproximations of the reachable set at the time $t = t_0+\delta, \ldots, t_0+N_a\delta$. The exact system execution which is denoted by the red dotted curve is guaranteed to be contained in the overapproximations at the discrete times.

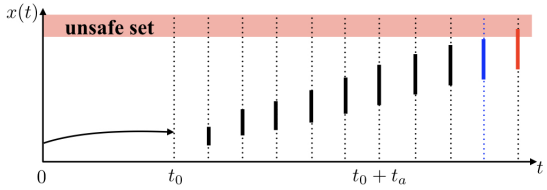


Fig. 8. Deadline estimation. Line segments: overapproximations of the reachable set at the time $t = t_0 + \delta, t_0 + 2\delta, \ldots$. The deadline is computed as $t = t_0 + t_a + 3\delta$.

as a conservative estimation for the start state in the recovery problem. The main idea is illustrated in Fig. 7.

Support function. The support function of a set $S \subseteq \mathbb{R}^n$ according to a vector $\vec{l}$ is defined by $\rho_S(\vec{l}) = \sup_{s\in S}\{\vec{l}^T s\}$. The support functions on convex sets have the following properties:

$$\begin{aligned} &\rho_{AS}(\vec{l}) = \rho_S(A^T\vec{l}), && \text{for convex set } S \\ &\rho_{S_1\oplus S_2}(\vec{l}) = \rho_{S_1}(\vec{l}) + \rho_{S_2}(\vec{l}) && \text{for convex sets } S_1, S_2 \end{aligned}$$

and for $\Omega_j = A^j\mathcal{B}_\epsilon$, we have that $\rho_{\Omega_j}(\vec{l}) = \sqrt{\vec{l}^T A^j (A^j)^T \vec{l}}\epsilon$, and hence we have the above two expressions for computing the upper and lower bounds respectively in the i-th dimension for X_0.

Computational cost for X_0. We assume that the system has n state variables and m inputs. We need to compute the lower and upper bounds in n dimensions for X_0. Although $\vec{x}_a$, and $\vec{c}_1, \ldots, \vec{c}_{N_a}$ can only be obtained online, we may precompute the matrices $\vec{l}^T A^k$, $\vec{l}^T A^k B$ and $\vec{l_i}^T A^k (A^k)^T \vec{l_i}\epsilon$ for some large number k offline. Hence, the most computational expensive work in computing each bound is the one multiplication of two n-dimensional vectors in computing $\vec{l_i}^T A^{N_a}\vec{x}_a$, and the N_a multiplications of two m-dimensional vectors in computing $\sum_{j=0}^{N_a-1} \vec{l_i}^T A^j B\vec{c}_{N_a-j}$.

C. Estimating a Conservative Deadline

Catastrophic events are described by unsafe state sets in the analysis of CPS, and it is essential to recover a system before any dangerous state is reached. However, computing the earliest time when a system reaches an unsafe condition is not easy since we do not know the actual system state after the sensor attack begins. Therefore, we seek to compute a time earlier than the exact deadline for recovering the system, and we will do so based on safety verification.

We compute the reachable sets for the LTI approximation model starting from the nearest trustworthy sensor data that is the state at the time t_0. The conservative deadline will be the time of the reachable set right before the first reachable set which has nonempty intersection with the unsafe set, as it is shown in Fig. 8.

Since a valid deadline should occur after the attack is detected which is at the time $t_0 + t_a$ and we assume that the system is safe during the time interval $[t_0, t_0 + t_a]$, we start the reachability computation from the set (4) which is denoted by Z_0, and the reachable set at the i-th step is

$$Z_i = \left\{ A^{N_a+i}\vec{x}_a + \sum_{j=i}^{N_a+i-1} A^j B\vec{c}_{N_a+i-j} \right\} \oplus \bigoplus_{j=0}^{N_a+i-1} A^j\mathcal{B}_\epsilon \oplus \underbrace{\left\{ \sum_{j=0}^{i-1} A^j B\vec{c}_{N_a+i-j} \right\}}_{\Phi_i}.$$

When the unsafe set is defined by linear constraints, we only need to iteratively compute the corresponding support functions for Z_i and check the safety until the first unsafe set is detected.

Computation of Φ_i. The first part of Z_i is known based on the nearest trustworthy state $\vec{x}_a$, the step number N_a between the beginning and being detected of the attack, and the number i. However, the future control inputs $\vec{c}_{N_a+1}, \ldots, \vec{c}_{N_a+i}$ which are needed in computing Φ_i are unknown when the attack is detected. To fill these unknown values in predicting the safety of Z_i, notice that our purpose is to find only a time such that it is possible for the controller to prevent the system from reaching any unsafe state before it, so using it is reasonable in our recovery task, therefore we may simply fill these control inputs by zero or the input used in the current step. We may also obtain two deadlines from both of the methods and choose the later one for our recovery work.

Safety checking based on support functions. Assume that the safe set X_S is defined by the set of states satisfying a conjunction of linear constraints $\bigwedge_{p=1}^{q} a_p^T\vec{x} \le b_p$, and the unsafe set is its complement. Then the reachable set Z_i is unsafe if and only if there is some $1 \le p \le q$ such that the upper bound of $a_p^T Z_i$ is greater than b_p, which can be known by verifying whether

$$\begin{aligned} &a_p^T A^{N_a+i}\vec{x}_a + \textstyle\sum_{j=0}^{N_a+i-1} a_p^T A^j B\vec{c}_{N_a+i-j} \\ &+ \textstyle\sum_{j=0}^{N_a+i-1} \sqrt{a_p^T A^j (A^j)^T a_p}\epsilon \end{aligned} \overset{?}{>} b.$$

Computational cost for Z_i. The analysis here is similar to that for X_0. The matrices $a_p^T A^k$, $a_p^T A^k B$ and $a_p^T A^k (A^k)^T a_p\epsilon$ for some large number k can also be computed offline in advance, because the LTI system and the unsafe set are known a prior. The most expensive task to check each safety constraint is to

perform one multiplication of two n-dimensional vectors in computing $a_p^T A^{N_a+i}\vec{x}_a$, and (N_a+i) multiplications for two m-dimensional vectors in computing $\sum_{j=0}^{N_a+i-1} a_p^T A^j B\vec{c}_{N_a-j}$.

It is possible to meet the following two extreme cases in finding a deadline. (a) The deadline obtained is exactly the current time. This could happen when the current system state is very close to or already in the unsafe set. If it is the case then our recovery task fails. (b) The estimation algorithm could not find a violation of the safe set after checking a large number of steps. This could happen if the current system state is far from the unsafe set and there is no danger over a long time. Since the deadline estimation is done online, it has to be efficient. To avoid checking too many reachable sets, which is unnecessary, we set up an upper limit $D_{\max}$ for the number of i, when there is no unsafe Z_i for all $1 \leq i \leq D_{\max}$, then $D_{\max}$ is used as the deadline.

Note that our recovery framework is not confined to certain deadline calculation approaches but always applicable as long as a safety deadline can be given, i.e., the deadline selected should be valid for just one control sequence.

VI. Evaluation

We implemented a prototype tool of our real-time recovery framework as well as the simulators for our CPS benchmarks in Python. We consider the following CPS models: vehicle turning, RLC circuit, DC motor position, aircraft pitch and quadrotor. The LTI model for the plant in each benchmark is obtained from first linearizing and then discretizing the original dynamics in a bounded state space which is sufficiently large to contain the CPS executions, and ϵ used in recovery is the overall error bound introduced by the LTI simplification.

For each CPS model, we consider three sensor attack scenarios: modification, delay and replay attacks. Our prototype tool is tested in the following way. For each scenario, we set a start time and a detection time for the sensor attack, the CPS execution is simulated by the simulator. When an attack is detected, our recovery controller is activated immediately to compute a recovery control sequence, and the result will be applied immediately to the simulator when it is obtained. Although our approach does not consider the time cost of computing a recovery control in theory, this runtime overhead is included in our experiments.

For each scenario, we provide a comparison among the following three system executions:

No recovery: the system run under the sensor attack without recovery. The system may drift to an unsafe state.

Non-real-time recovery: the system run under the recovery method presented in [9]. This method does not guarantee the recovery deadline.

Real-time recovery: the system run under the real-time recovery approach presented in the paper.

The experimental results are presented in Fig. 9 and Table I. In addition, we also evaluate the scalability of our approach based on the study of a model of heating in a point of a rod located at $1/3$ of the length and recording the temperature at $2/3$ of the length [54]. The model can be described by a linear differential equation with $N \geq 2$ variables each of which denotes the temperature in a position on the rod. With more state variables, we have a more accurate model. The details are given in the rest of the section.

We provide a detailed description of all intermediate computation results in the first benchmark, but only give the experimental settings and final results for the rest of the tests.

Vehicle turning. Our first case study is a simple system that models the turning of a vehicle changing the speed of each wheel differently [9]. The physical dynamics are modeled by the ODE

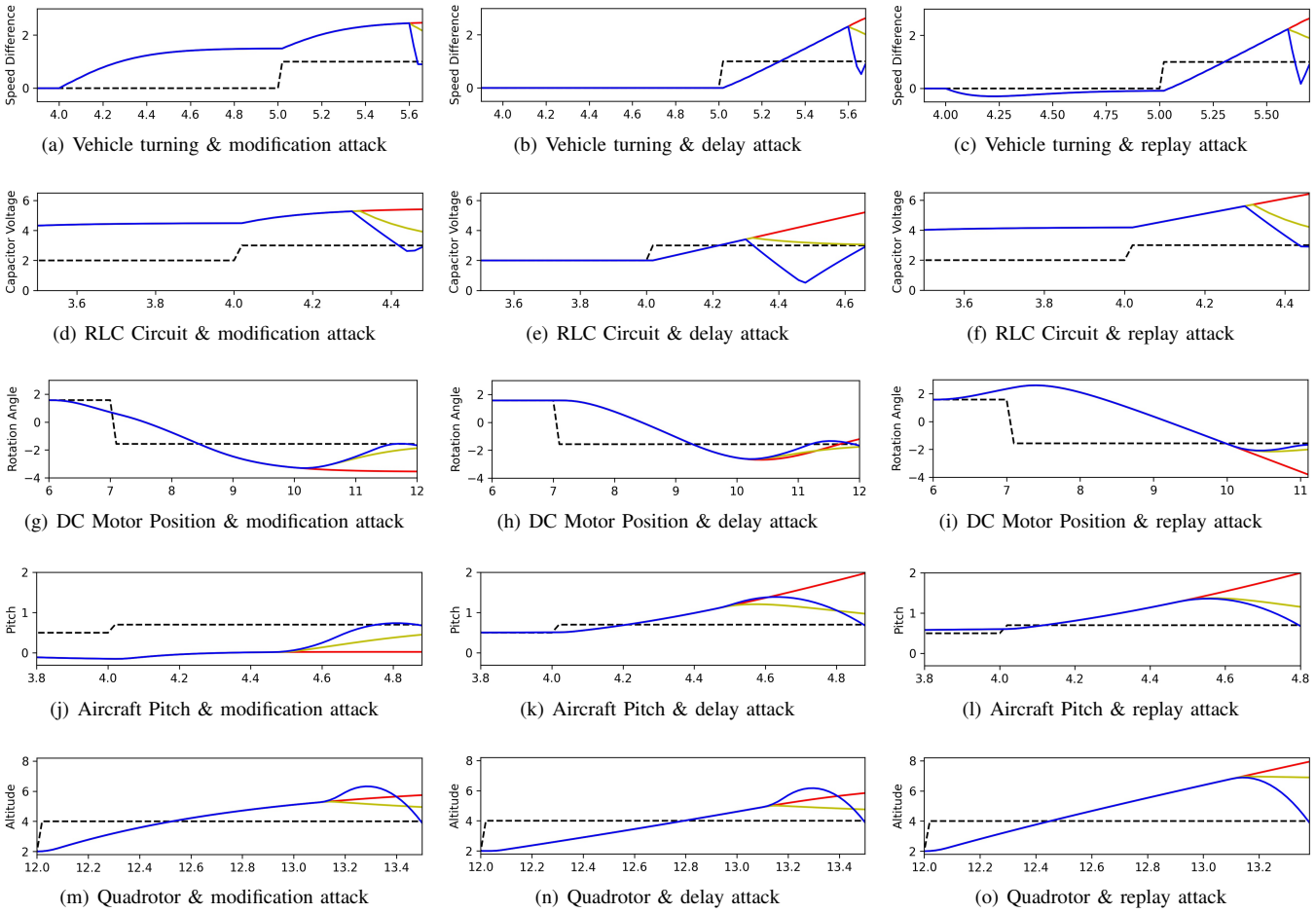
$$\dot{x} = -\frac{25}{3}x + 5u$$

where x denotes the speed difference of the wheels and u is the control input which is the difference of the voltages applied to the motors controlling the two wheels. When the vehicle is moving straight, both of x and u are zero. When the car is turning right, the left wheel should be rotating faster than the right one. Here, the value of u is updated by the PID controller every 0.02 seconds with the coefficients $K_P = 0.5$ and $K_I = 7$ to maintain the speed difference to a reference value which is 1 meter per second here. We consider the behavior of the system start from the state $x = 1$, such that the controller initially tries to make the vehicle move straight, i.e., steering the value of x to 0, and at the time $t = 5$ seconds, turns the vehicle to right, i.e., steering the value of x to 1. The performance of our recovery approach is evaluated in the following attack scenarios.

Modification attack. The attack adds a value of -1.5 to the sensor data sent to the controller in every step starting at time $t = 4$, and we assume that it is detected at time $t = 5.6$ and our recovery approach tries to steer the system to the target set $x \in [0.8, 1.2]$ while avoiding reaching anywhere outside of the safe set which is $x \in [-2.75, 2.75]$. We use a linear discretization of the system ODE with the error bound $\epsilon < 10^{-7}$. As suggested by Theorem IV.1 and IV.2, we conservatively shrink the safe set to $x \in [-2.7, 2.7]$ and target set to $x \in [0.9, 1.1]$. The recovery start state estimation X_0, i.e., the conservative estimation of the system state at the time $t = 5.6$, and the deadline is computed as 3 control steps. As shown in Fig. 9(a), our approach successfully recovery the system before the deadline while the method in [9] is not able to do that.

Delay attack. The delay attack delays the transmission of the sensor data to the controller by 1 second. We consider the same start time, the time of detection, the safe set and the target set as those in the modification attack scenario. The estimation for X_0 is the range of $[2.31251209, 2.31251339]$, and the deadline is estimated as 4 steps. A comparison of the system executions is shown in Fig. 9(b).

Replay attack. The attacker replays the sensor data from the time 0 to 6 with the other settings same as those in the modification attack scenario. The estimation X_0 is obtained as $[2.2345892, 2.23459051]$, and deadline is computed as 5 steps. Fig. 9(c) shows our recovery result.



(a) Vehicle turning & modification attack (b) Vehicle turning & delay attack (c) Vehicle turning & replay attack

(d) RLC Circuit & modification attack (e) RLC Circuit & delay attack (f) RLC Circuit & replay attack

(g) DC Motor Position & modification attack (h) DC Motor Position & delay attack (i) DC Motor Position & replay attack

(j) Aircraft Pitch & modification attack (k) Aircraft Pitch & delay attack (l) Aircraft Pitch & replay attack

(m) Quadrotor & modification attack (n) Quadrotor & delay attack (o) Quadrotor & replay attack

Fig. 9. Comparison of the system executions under three situations for each attack scenario. RED = No recovery. YELLOW = Non-real-time recovery (previous work [9]). BLUE = Real-time recovery (our proposal). Dotted Black Line = Reference state.

Series RLC Circuit. We consider the model of a basic RLC circuit consisting of a resistor, an inductor, and a capacitor connected in series. The state of the model is described by two variables x_1: the voltage of the capacitor, and x_2: the current in the loop. The control input u is considered as the voltage applied to the inductor. The dynamics of the system are given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u.$$

We assume that the voltage u is updated by a PI controller every 0.02 seconds with the coefficients $K_P = 5$ and $K_I = 5$, and it tries to maintain the voltage across the capacitor to a reference value. Similar to the previous benchmark, we consider three attack scenarios and use the start time $t = 3$ and the time of detection $t = 4.3$ in all of the scenarios. The system is safe when the voltage x_1 is between $[0, 7]$, and the target set under our consideration in all attack scenarios is $[2.9, 3.1]$ for x_1.

Modification attack. The attacker subtracts the value 2.5 from the sensor data sent to the controller.

Delay attack. The attacker delays the sensor data by 1 second, that is 50 control steps.

Replay attack. The attacker replays the sensor data in the first 250 steps.

Fig. 9(d), 9(e), and 9(f) show the experimental results. In Fig. 9(e), our real-time recovery control firstly steers the capacitor voltage to a low value and then drives it back to a value close to the reference before the deadline. This phenomenon can also been seen in some of the other tests and is caused by the LP solver which searches the control inputs along the boundary of the solution set. However, the recovery control still safely steers the system to a target state.

DC Motor Position. The following ODE describes the control of the position of a DC motor.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{b}{J} & \frac{K}{J} \\ 0 & -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} u$$

where x_1 is the rotation angle, x_2 is the angular velocity, and x_3 denotes the armature current. The other parameters are the same as those in Example. III.1. The PD controller with coefficients $K_P = 11$ and $K_D = 5$ updates the value of

u every 0.1 second to maintain the rotation angle x_1 along a reference angle. The start time in all attack scenarios is $t = 6$ and the detection time is 4 seconds later. The DC motor is considered to be safe if x_1 is between $[-4, 4]$ while the target set is $x_1 \in [-1.67, -1.47]$.

Modification attack. The attacker adds the value 2 to the sensor data transmitted to the controller.

Delay attack. The attacker delays the sensor data by 1 second, that is 10 control steps.

Replay attack. The attacker replays the sensor data in the first 6 seconds.

Aircraft Pitch. The model of aircraft pitch can be described by the following 3-dimensional ODE

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix} u$$

such that x_1 denotes the angle of attack, x_2 denotes the pitch rate, and x_3 denotes the pitch angle. The control input is the variable u which is the elevator deflection angle. We use a PID controller with the coefficients $K_p = 14, K_i = 0.8, K_d = 5.7$ which updates u every 0.02 second to maintain the value of x_3 along a reference angle. The start time in all attack scenarios is $t = 3$ and the time of detection is $t = 4.46$. The system is safe when the pitch angle x_3 is between $[0, 2]$, and the target set is $x_3 \in [0.68, 0.72]$.

Modification attack. The attacker adds 0.68 to the sensor data that transmitted to the controller.

Delay attack. The attacker delays the sensor data by 1 second.

Replay attack. The attacker replays the sensor data in the time interval of $[1, 2]$.

Quadrotor. We consider a linear quadrotor model described in [55]. The system consists of 12 state variables: (x, y, z) denotes the (relative) position, (ϕ, θ, ψ) denotes the angles of pitch, yaw and roll respectively, (u, v, w) and (p, q, r) are the velocity and angular velocity of the quadrotor. The controller produces 4 inputs which are f_t: total thrust, and $[\tau_x, \tau_y, \tau_z]^T$: control torques caused by differences of rotor speeds. We consider a discrete PD controller with the coefficients $K_P = 0.1$ and $K_D = 0.6$ to maintain the altitude of the quadrotor at $z = 4$ by a time stepsize of 0.02 seconds. The attack scenarios are given as below. The start time in all scenarios is $t = 12$ and the detection time is 1.1 seconds later. The quadrotor is considered to be safe if its altitude is between $[-1, 8]$. We use the target set $[3.9, 4.1]$ for the altitude.

Modification attack. The attacker subtracts all values from the sensor data by 2. *Delay attack.* The attacker delays the sensor data by 1 second. *Replay attack.* The attacker starts to replay the sensor data from $t = 0$ to 5.

Similar to most of the formal verification and synthesis techniques on CPS, our real-time recovery approach also assumes that a set of control inputs can be calculated immediately based on the current sensor data. The reason to do so is the hardness of knowing the time cost of a controller due to the highly uncertain hardware performance. However such an assumption can never be perfectly achieved in practice. Therefore, in Table I, we present the intermediate time costs as well as the total time cost of computing a recovery control for each attack scenario. We also show the ratios of the time cost to the control stepsize, and most of them are below 50%. As shown by the simulation results, all of the scenarios are successfully recovered by our method.

Scalability Analysis. We investigate the scalability of our approach based on a scalable heating model which is presented in [54]. The model describes heating in a point of a rod located at $1/3$ of the length and recording the temperature at $2/3$ of the length. The temperatures of the selected points on the rod is described by a linear differential equation of the form $\dot{\vec{x}} = A\vec{x} + Bu$ such that

$$A = \frac{\alpha}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

B is a column vector whose $(n/3)$-th component is 1 and others are zero, where n is the scale of A. We consider the scales $n = 25, 30, 35, 40, 45$ and design a PID controller whose stepsize is 0.2 seconds for each of them to heat the point at $1/3$ of the length from 0°C to 15°C, and produces similar behaviors. Since our focus is on the scalability test, we only consider one attack scenario in which the attack starts to subtract the recorded temperature value at $2/3$ of the length by 50 at the time $t = 0$, and it is detected at the time $t = 20$. Our recovery task is to drive the temperature at $1/3$ of the length to the target degree range $[7, 25]$ within the safe temperature range $[0, 40]$. We bound the maximum recovery steps by 10. Table II shows the intermediate and the total time costs for computing the recovery control sequences. It can be seen that even for 45 state variables, our method still costs only $\sim 50\%$ of the time period of the control stepsize to obtain a 10-step recovery control.

VII. Conclusions

In this paper, we proposed a new attack-response architecture, and a set of algorithms supported by formal methods to perform a timely and safe recovery of a CPS after an attack has been detected. We provided a formal analysis of our algorithms, and in addition we evaluated the performance based on various attack scenarios and also performed a scalability test. The use-cases we analyze, show how our methods are able to recover the attacked-system in a timely and safe manner, outperforming previous related work.

There are several possible extensions for future work. For example, in our analysis and simulations, we assumed a worst-case scenario where all sensors were compromised, but our system can perform better if only a subset of the sensors is compromised, and the intrusion detection system tells us which specific sensors are the ones we need to remove. Another possible extension is our work on reachability analysis. We will investigate the use of Linear Time-Varying models in

TABLE I

TIME COSTS OF COMPUTING THE RECOVERY CONTROLS. THE TIME UNIT IS MILLISECOND. LEGENDS: #1: VEHICLE TURNING, #2: RLC CIRCUIT, #3: DC MOTOR POSITION, #4: AIRCRAFT PITCH, #5: QUADROTOR, A: ATTACK TYPE, δ: CONTROL STEPSIZE, k: # OF STEPS IN THE RECOVERY CONTROL, X_0: TIME COST OF ESTIMATING X_0, T_D: TIME COST OF CALCULATING THE DEADLINE, T_F: TIME COST OF FORMULATING THE LP PROBLEM, T_S: TIME COST OF SOLVING THE LP PROBLEM, %: RATIO OF THE TOTAL TIME COST TO THE TIME PERIOD OF A CONTROL STEPSIZE.

B	δ	A	k	X_0	T_D	T_F	T_S	Total	%
#1	20	M	3	0.35	0.29	0.07	0.03	0.74	3.71%
		D	4	0.34	0.35	0.06	0.02	0.77	3.84%
		R	5	0.34	0.41	0.07	0.03	0.85	4.24%
#2	20	M	9	0.34	0.67	0.22	0.07	1.30	6.52%
		D	18	0.34	1.62	0.41	0.14	2.49	12.46%
		R	8	0.31	0.65	0.12	0.06	1.14	5.69%
#3	100	M	20	0.53	1.59	1.00	0.28	3.40	3.40%
		D	20	0.28	1.54	1.70	0.29	3.81	3.81%
		R	11	0.33	0.90	0.41	0.12	1.76	1.76%
#4	20	M	21	0.34	2.02	0.97	0.31	3.64	18.21%
		D	21	0.36	2.02	1.50	0.29	4.17	20.86%
		R	17	0.35	1.43	0.75	0.21	2.74	13.69%
#5	20	M	20	0.53	1.81	7.52	1.14	11.0	55.01%
		D	20	0.43	1.75	7.38	1.14	10.70	53.55%
		R	14	0.50	1.48	3.49	0.59	6.06	30.28%

TABLE II

SCALABILITY EVALUATION. THE TIME UNIT IS MILLISECOND. LEGENDS: n: # OF VARIABLES, OTHERS: SAME AS THOSE IN TABLE I.

n	X_0	T_D	T_F	T_S	Total	%
20	0.57	0.96	17.37	4.99	23.89	11.94%
25	0.57	0.99	41.26	6.95	49.77	24.88%
30	0.63	1.03	59.59	8.00	69.25	34.62%
35	0.66	1.11	74.64	10.22	86.63	43.32%
40	0.74	1.17	81.77	13.15	96.83	48.42%
45	0.75	1.28	86.68	17.23	105.94	52.97%

our recovery framework. Such models are able to better approximate nonlinear dynamics even in a large state space.

Attack detection and recovery are essential for maintaining safety and resilience in cyber-physical systems. While the work on attack *detection* is extensive, work on attack *recovery* has received comparatively little attention. In order to fully realize the promise of various intrusion detection efforts, we need to provide safe and secure attack-response mechanisms that are activated automatically after an alert. We hope that our work will motivate more research in this area.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive comments and thank the anonymous shepherd for being with us along the revision process. This work was supported in part by NSF CCF-1720579, NSF CCF-2028740, NSF CNS-1931573 and the U.S. Air Force Research Laboratory (AFRL) under contract number FA8650-16-C-2642.

REFERENCES

[1] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Design Automation Conference (DAC)*. IEEE, 2010, pp. 731–736.

[2] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 163–168.

[3] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, 2016.

[4] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *The 28th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 2008, pp. 495–500.

[5] N. Adam, "Workshop on future directions in cyber-physical systems security," in *Report on workshop organized by Department of Homeland Security (DHS)*, 2010.

[6] M. Wolf and D. Serpanos, "Safety and security in cyber-physical systems and internet-of-things systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 9–20, 2017.

[7] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, "Non-invasive spoofing attacks for anti-lock braking systems," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2013, pp. 55–72.

[8] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, p. 2015, 2015.

[9] F. Kong, M. Xu, J. Weimer, O. Sokolsky, and I. Lee, "Cyber-physical system checkpointing and recovery," in *ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 22–31.

[10] F. Kong, O. Sokolsky, J. Weimer, and I. Lee, "State consistencies for cyber-physical system recovery," in *Workshop on Cyber-Physical Systems Security and Resilience (CPS-SR)*, 2019.

[11] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. J. Pappas, "Robustness of attack-resilient state estimators," in *ACM/IEEE 5th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE Computer Society, 2014, pp. 163–174.

[12] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 801–816.

[13] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "SAVIOR: Securing autonomous vehicles with robust physical invariants," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[14] R. Ivanov, M. Pajic, and I. Lee, "Attack-resilient sensor fusion for safety-critical cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 1, p. 21, 2016.

[15] B. Ao, Y. Wang, L. Yu, R. R. Brooks, and S. Iyengar, "On precision bound of distributed fault-tolerant sensor fusion algorithms," *ACM Computing Surveys (CSUR)*, vol. 49, no. 1, p. 5, 2016.

[16] P. Lu, L. Zhang, B. B. Park, and L. Feng, "Attack-resilient sensor fusion for cooperative adaptive cruise control," in *21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3955–3960.

[17] M. Pajic, J. Weimer, N. Bezzo, O. Sokolsky, G. J. Pappas, and I. Lee, "Design and implementation of attack-resilient cyberphysical systems: With a focus on attack-resilient state estimators," *IEEE Control Systems Magazine*, vol. 37, no. 2, pp. 66–81, 2017.

[18] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[19] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–29, 2014.

[20] P. Guo, H. Kim, N. Virani, J. Xu, M. Zhu, and P. Liu, "Roboads: Anomaly detection against sensor and actuator misbehaviors in mobile robots," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 574–585.

[21] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2016, pp. 130–139.

[22] A. Ganesan, J. Rao, and K. Shin, "Exploiting consistency among heterogeneous sensors for vehicle anomaly detection," SAE Technical Paper, Tech. Rep., 2017.

[23] T. He, L. Zhang, F. Kong, and A. Salekin, "Exploring inherent sensor redundancy for automotive anomaly detection," *57th Design Automation Conference*, 2020.

[24] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *2010 Sixth International Conference on Information Assurance and Security*. IEEE, 2010, pp. 92–98.

[25] J. Giraldo, E. Sarkar, A. A. Cardenas, M. Maniatakos, and M. Kantarcioglu, "Security and privacy in cyber-physical systems: A survey of surveys," *IEEE Design & Test*, vol. 34, no. 4, pp. 7–17, 2017.

[26] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE transactions on control systems technology*, vol. 18, no. 3, pp. 636–653, 2009.

[27] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, p. 13, 2011.

[28] A. A. Cardenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proceedings of the 6th ACM symposium on information, computer and communications security*, 2011, pp. 355–366.

[29] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the plc: semantic security monitoring for industrial processes," in *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, 2014, pp. 126–135.

[30] Y. Chen, C. M. Poskitt, and J. Sun, "Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system," *IEEE Symposium on Security and Privacy*, 2018.

[31] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the impact of stealthy attacks on industrial control systems," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1092–1105.

[32] K. Paridari, N. O'Mahony, A. E.-D. Mady, R. Chabukswar, M. Boubekeur, and H. Sandberg, "A framework for attack-resilient industrial control systems: Attack detection and controller reconfiguration," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 113–128, 2018.

[33] R. Ma, S. Basumallik, S. Eftekharnejad, and F. Kong, "Recovery-based model predictive control for cascade mitigation under cyber-physical attacks," in *2020 IEEE Texas Power and Energy Conference (TPEC)*. IEEE, 2020, pp. 1–6.

[34] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-time systems*, vol. 28, no. 2-3, pp. 101–155, 2004.

[35] T. T. Johnson, S. Bak, M. Caccamo, and L. Sha, "Real-time reachability for verified simplex design," *ACM Trans. Embedd. Comput. Syst.*, vol. 15, no. 2, p. 29, May 2016.

[36] X. Chen and S. Sankaranarayanan, "Decomposed reachability analysis for nonlinear systems," in *2016 IEEE Real-Time Systems Symposium (RTSS)*. IEEE Press, Nov 2016, pp. 13–24.

[37] ——, "Model-predictive real-time monitoring of linear systems," in *IEEE Real-Time Systems Symposium (RTSS)*. IEEE Press, 2017, pp. 297–306.

[38] H. Yoon, Y. Chou, X. Chen, E. Frew, and S. Sankaranarayanan, "Predictive runtime monitoring for linear stochastic systems and applications to geofence enforcement for UAVs," in *Proceedings of Runtime Verification 2019*, ser. Lecture Notes in Computer Science, B. Finkbeiner and L. Mariani, Eds., vol. 11757. Springer, 2019, pp. 349–367.

[39] C. Le Guernic and A. Girard, "Reachability analysis of linear systems using support functions," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 250 – 262, 2010.

[40] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV)*, ser. Lecture Notes in Computer Science, vol. 6806. Springer, 2011, pp. 379–395.

[41] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Correct, reactive, high-level robot control," *IEEE Robotics Automation Magazine*, vol. 18, no. 3, pp. 65–74, 2011.

[42] S. Magdici and M. Althoff, "Adaptive cruise control with safety guarantees for autonomous vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5774 – 5781, 2017, 20th IFAC World Congress.

[43] C. Fan, U. Mathur, S. Mitra, and M. Viswanathan, "Controller synthesis made real: Reach-avoid specifications and linear dynamics," in *Proceedings of the 30th International Conference on Computer Aided Verification (CAV)*, ser. Lecture Notes in Computer Science, vol. 10981. Springer, 2018, pp. 347–366.

[44] T. Kushner, D. M. Bortz, D. M. Maahs, and S. Sankaranarayanan, "A data-driven approach to artificial pancreas verification and synthesis," in *9th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. IEEE Computer Society / ACM, 2018, pp. 242–252.

[45] B. Lu, "Linear parameter-varying control of an f-16 aircraft at high angle of attack," Ph.D. dissertation, North Carolina State University, 2004.

[46] M. Sato, "Robust model-following controller design for LTI systems affected by parametric uncertainties: a design example for aircraft motion," *International Journal of Control*, vol. 82, no. 4, pp. 689–704, 2009.

[47] I. Jovanov and M. Pajic, "Relaxing integrity requirements for attack-resilient cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 4843–4858, 2019.

[48] A. Khazraei and M. Pajic, "Attack-resilient state estimation with intermittent data authentication," 2020.

[49] E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Inf.*, vol. 43, no. 7, pp. 451–476, 2007.

[50] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization," in *Proceedings of the 47th IEEE Conference on Decision and Control (CDC)*. IEEE, 2008, pp. 4042–4048.

[51] T. Dang, O. Maler, and R. Testylier, "Accurate hybridization of nonlinear systems," in *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2010, pp. 11–20.

[52] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo, "S3a: secure system simplex architecture for enhanced security of cyber-physical systems," *arXiv preprint arXiv:1202.5722*, 2012.

[53] C. Le Guernic, "Reachability analysis of hybrid systems with linear continuous dynamics," Ph.D. dissertation, Université Joseph Fourier, 2009.

[54] Y. Chahlaoui and P. Van Dooren, "A collection of benchmark examples for model reduction of linear time invariant dynamical systems," *SLICOT Working Note*, 2002.

[55] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation," Master's thesis, KTH Royal Institute of Technology, 2015.