A Timing-Based Framework for Designing Resilient Cyber-Physical Systems under Safety Constraint

ABDULLAH AL MARUF and LUYAO NIU, Network Security Lab, Department of Electrical and Computer Engineering, University of Washington, USA
ANDREW CLARK, Electrical and Systems Engineering Department, McKelvey School of Engineering, Washington University in St. Louis, USA
J. SUKARNO MERTOGUNO, School of Cybersecurity and Privacy, Georgia Institute of Technology, USA
RADHA POOVENDRAN, Network Security Lab, Department of Electrical and Computer Engineering, University of Washington, USA

Cyber-physical systems (CPS) are required to satisfy safety constraints in various application domains such as robotics, industrial manufacturing systems, and power systems. Faults and cyber attacks have been shown to cause safety violations, which can damage the system and endanger human lives. Resilient architectures have been proposed to ensure safety of CPS under such faults and attacks via methodologies including redundancy and restarting from safe operating conditions. The existing resilient architectures for CPS utilize different mechanisms to guarantee safety, and currently, there is no common framework to compare them. Moreover, the analysis and design undertaken for CPS employing one architecture is not readily extendable to another. In this article, we propose a timing-based framework for CPS employing various resilient architectures and develop a common methodology for safety analysis and computation of control policies and design parameters. Using the insight that the cyber subsystem operates in one out of a finite number of statuses, we first develop a hybrid system model that captures CPS adopting any of these architectures. Based on the hybrid system, we formulate the problem of joint computation of control policies and associated timing parameters for CPS to satisfy a given safety constraint and derive sufficient conditions for the solution. Utilizing the derived conditions, we provide an algorithm to compute control policies and timing parameters relevant to the employed architecture. We also note that our solution can be applied to a wide class of CPS with polynomial dynamics and also allows incorporation of new architectures. We verify our proposed framework by performing a case study on adaptive cruise control of vehicles.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; **Dependable and fault-tolerant systems and networks;**

This work was supported by the AFOSR grants FA9550-20-1-0074 and FA9550-22-1-0054, by the Office of Naval Research grants N00014-20-1-2636 and N00014-23-1-2386, by the National Science Foundation grant CNS-1941670, and by the BIRD Foundation: Israel-US Energy Center, Cyber Topic.
Authors' addresses: A. Al Maruf, L. Niu, and R. Poovendran, Network Security Lab, Department of Electrical and Computer Engineering, University of Washington, P.O. Box 1212, Seattle, Washington 98195-2500; emails: {maruf3e, luyaoniu, rp3}@uw.edu; A. Clark, Electrical and Systems Engineering Department, McKelvey School of Engineering, Washington University in St. Louis, Missouri 63130; email: andrewclark@wustl.edu; J. Sukarno Mertoguno, School of Cybersecurity and Privacy, Georgia Institute of Technology, Atlanta, Georgia 30332; email: karno@gatech.edu.

2378-962X/2023/07-ART19 $15.00
https://doi.org/10.1145/3594638

Additional Key Words and Phrases: Safety-critical system, barrier certificate

ACM Reference format:
Abdullah Al Maruf, Luyao Niu, Andrew Clark, J. Sukarno Mertoguno, and Radha Poovendran. 2023. A Timing-Based Framework for Designing Resilient Cyber-Physical Systems under Safety Constraint. *ACM Trans. Cyber-Phys. Syst.* 7, 3, Article 19 (July 2023), 25 pages.
https://doi.org/10.1145/3594638

1 INTRODUCTION

The coupling between cyber and physical subsystems of **Cyber-Physical Systems** (**CPS**) creates an opportunity for faults and attacks on cyber components to impact the physical performance and safety [4, 22]. Cyber attacks causing severe safety violations have been reported in many application domains such as transportation [16], medical devices [18], and power system [24].

A variety of fault tolerant architectures [9, 29, 41] have been proposed to mitigate the impact of failures in CPS. These architectures primarily use redundancies and require at least one of the redundant components to be verified as fault-free at any time instant. However, when CPS are under cyber attacks, these architectures become sub-optimal or even not applicable since an intelligent adversary can exploit the vulnerabilities common to all the components and cause system failure.

To address cyber attacks against CPS, **cyber resilient architectures** (**CRAs**) [1, 2, 8, 27, 32, 38, 39] have been proposed to ensure the safety of CPS during attacks and recover the system to its nominal operation after attacks. One class of architectures mitigates cyber attacks by restarting the cyber subsystem to an uncompromised ("clean") state [1, 8, 39]. This restart can either be proactive (periodic) or reactive by engineering the controller to crash when the adversary attempts to exploit the vulnerabilities. Such restart-based mechanisms limit the adversary's capabilities of compromising the cyber component, but also render the controller inoperative during restart, potentially leading to safety violations.

The existing CRAs [1, 8, 15, 27, 32, 39] employ different mechanisms that achieve different levels of resilience, safety, and system performance. At present, there is no common analysis methodology to evaluate them or compute the associated design parameters and control policies for CPS safety. Such a methodology should be based on a framework to model CPS employing all possible CRAs, which currently does not exist.

In this article, we propose and develop a hybrid system framework to provide guarantee on safe operation of a CPS that uses different classes of CRAs. Our insight is that the cyber subsystem operates in one out of a finite number of statuses, such as *normal* (following nominal control policy), *corrupted* (controller compromised by the adversary), and *restoration* (recovering the controller). These statuses comprise the discrete location set of the hybrid system, whereas the physical dynamics of the CPS are captured by a nonlinear dynamical state space model. Based on the hybrid system, we develop a common analysis methodology to compute control policies and associated design parameters for different classes of CRAs. Our approach is based on computing the duration of time that the CPS can remain in each status to guarantee CPS safety. These time durations, which we denote as *timing parameters*, provide a common method to quantify the resilience of CPS when employing distinct CRAs, and thus allow us to evaluate the CRAs. Our approach is sufficiently general to enable analysis and design for future CRAs. Our main contributions in this article are summarized as follows.

— We develop a hybrid system model to capture CPS employing at least ten state-of-the-art architectures of six different classes [1, 7–9, 15, 27, 29, 32, 39, 41]. The discrete transitions of the hybrid system model how cyber statuses evolve over time following cyber attacks.

— Using the hybrid system, we quantify how fast CPS approaches the boundary of a given safety region at each cyber status. Using this quantification, we derive sufficient conditions for a control policy and timing parameters so that the system remains within the safety set.
— We present an algorithm to jointly compute the control policy and timing parameters by mapping the derived conditions to a **sum-of-squares** (**SOS**) program that is applicable to any CPS with polynomial dynamics. We analyze our proposed algorithm and prove that it finds a solution, provided that one exists and satisfies certain constraints strictly.
— We validate our algorithm by using a case study on **adaptive cruise control** (**ACC**) of vehicles. We show that our approach guarantees safety of the system implementing any of the CRAs.

A preliminary version of this work was presented in Reference [31]. Compared with Reference [31], this article differs in the following aspects. First, our hybrid model incorporates more CRAs than [31], including proactive restart [1, 39], reactive restart [32], and dual redundant architectures [15]. Moreover, the conditions and algorithm derived in this article are applicable to CPS with a broader class of physical dynamics. Our proposed approach in this article, compared to the preliminary work is also more flexible since it can be mapped to new architectures, allowing future resilient designs to be incorporated into the framework. We present a new case study for which the solution method given in Reference [31] is not applicable.

The remainder of this article is organized as follows. Section 2 presents the related works. Section 3 presents the system and threat models. Our proposed framework and problem statement are presented in Section 4. Section 5 provides our solution approach including proposed algorithm and its analysis. A case study is presented in Section 6. We conclude the article in Section 7.

2 RELATED WORKS

There has been extensive study on verification [33, 36] and control synthesis [5, 13, 19, 37] for safety-critical CPS, assuming that the systems are operated under benign environments without any fault or attack. Barrier certificate based approaches, where safety constraint is encoded as a linear inequality on the control input, have been widely used in this context [6, 36, 46].

To address faults occurring in CPS, many fault tolerant controls [42, 45, 47] and architectures [9, 12, 29, 30, 41] have been proposed. Fault tolerant controls usually consist of fault detection scheme accompanied with resynthesis of controllers using methods like robust H_∞ control and model predictive control [47]. Fault tolerant architectures are primarily redundancy based and designed to deal with known failures which occur randomly.

One of the widely adopted fault tolerant designs is Simplex architecture [41]. This architecture consists of a main controller and a safety controller. The main controller is a high performance controller which is vulnerable to random failure whereas safety controller is verifiable and fault-free. Both controllers run in parallel and a decision module monitors the system states and decides which controller to be used to actuate the physical subsystem. The main controller operates the system unless the decision module instantly switches to safety controller under certain conditions, e.g., main controller is faulty. After recovering the main controller from fault, the decision module switches back to the main controller again. System-level Simplex [9] requires the safety controller and decision module to be located in a dedicated trusted processing unit such as FPGA. **Secure System Simplex** (**S3A**) [29] is an enhancement of system-level Simplex as the decision module also monitors the side channels of main controller for faster detection of certain cyber attacks that cannot replicate the monitored side channels. These fault tolerant architectures rely on the assumption that there is no common mode failure for all the controllers. However, this assumption may not hold in the presence of cyber attack [27].

In the existing literature, two different directions have been taken to address cyber attacks in CPS. In the first direction, control- and game-theoretic approaches have been proposed to protect CPS from cyber attacks [11, 14, 25, 28, 34, 48]. The idea is to detect the attack and then prevent the impact from the attack by making necessary corrections for the detected attack. The second direction focuses on designing resilient systems that can tolerate and then recover from attack. The CRAs [1, 7, 8, 15, 27, 32, 39], including **Byzantine fault tolerant++** (**BFT++**) [27] and **You Only Live Once** (**YOLO**) [8] follow this approach.

BFT++ [27] relies on redundancies of the controller. In particular, BFT++ uses multiple redundant controllers where one of them is designated as backup. Other non-backup controllers employ artificial software diversity via different implementations of software or randomization of the memory or instruction set [20, 23]. Due to this diversity, one of the non-backup controller deliberately crashes when an attacker attempts to intrude the system by exploiting the cyber vulnerabilities. This crash signal triggers restoration of non-backup controllers from the backup one.

YOLO and its variant use periodic restart and does not have any redundant controller [7, 8]. During each restart, the controller is reset to its "clean" state by loading its software from a read only module and clearing out all the volatile memory. YOLO also implements software diversity after each restart to ensure that the attacker cannot exploit same vulnerabilities. YOLO requires proper tuning of controller availability based on the natural resilience of the physical subsystem. Dual redundant scheme adopts same strategy as YOLO, but instead of restarting CPS, it periodically switches between two identical controllers [15]. After switching to the standby controller, the other controller restarts to ensure its integrity. This scheme is extendable to multiple controllers where CPS switches among them periodically. However, this redundant scheme is useful over YOLO only when the restart time of controllers is relatively high.

Proactive restart based scheme also uses restart to recover the compromised controller [1, 39]. Unlike YOLO, a secured execution interval (SEI) program is executed following a restart. During this interval, all the external interfaces of the system are disabled while a safety controller takes over the system. After scheduling the next restart time, the main controller program takes over from the safety controller and CPS resumes its normal operation by enabling the external interfaces. In Reference [1] online reachability analysis was used to determine the time for next restart and then it was scheduled in an external hardware timer which can only be programmed once before the next restart. Reference [39] provided a method to compute these timing parameters using offline reachability analysis over linearized dynamics of the physical subsystem. These results has been extended for networked systems as well as noisy settings [17, 40].

Reactive restart based mechanism uses crash signals as triggers to restart [32]. It assumed that the adversary needs certain amount of time (referred as exploitation window) to exploit the vulnerabilities to own the controller. The controller will then crash after some time (referred as vulnerability window) due to erroneous inputs coming from the adversary. Reference [32] provides an analysis using barrier certificate based approach to compute the control policy and bounds on these timing windows. However, this analysis does not hold for a system whose barrier certificate is of higher relative degree with respect to the dynamics of the physical subsystem.

3 SYSTEM AND THREAT MODEL

In this section, we first describe the system model. We then present the threat model.

3.1 System Model

Throughout this article, we use $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{R}_{>0}$, $\mathbb{Z}$, $\mathbb{Z}_{\geq 0}$, and $\mathbb{Z}_{>0}$ to denote the sets of real numbers, non-negative real numbers, positive real numbers, integers, non-negative integers, and positive integers, respectively. Given a vector $x \in \mathbb{R}^n$, we denote its ith entry as $[x]_i$, where $i = 1, \ldots, n$.

We consider a CPS comprised of a cyber subsystem and a physical subsystem. The physical subsystem is modeled by a plant that has the dynamics

$$\dot{x}_t = f(x_t) + g(x_t)u_t, \tag{1}$$

where $x_t \in \mathcal{X} \subset \mathbb{R}^n$ is the system state and $u_t \in \mathcal{U} \subset \mathbb{R}^m$ is the control input. Functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are assumed to be Lipschitz continuous. We also assume that the input set satisfies $\mathcal{U} = \prod_{i=1}^{m}[u_i^{min}, u_i^{max}]$ with $u_i^{min} \leq u_i^{max}$. A control policy $\mu : \mathcal{X} \to \mathcal{U}$ determines the actuator signal u_t given the system state x_t, so that $u_t = \mu(x_t) \in \mathcal{U}$ for $x_t \in \mathcal{X}$.

Although the physical plant evolves in continuous time, the cyber subsystem interacts with the physical subsystem at discrete instants of time. That is, the cyber subsystem reads the measurements of the physical subsystem's state and issues control command at discrete time $k\delta$, where $k \in \mathbb{Z}_{\geq 0}$ and $\delta \in \mathbb{R}_{>0}$. We refer to each discrete time interval of duration δ as an *epoch*. The duration δ of each epoch is dependent on the controller's sampling period.

Safety-critical CPS are required to operate within a certain range called as *safety set*. Here, we assume that the safety set of our considered CPS is given as $C = \{x \in \mathcal{X} : h(x) \geq 0\}$, where $h : \mathbb{R}^n \to \mathbb{R}$ is a rth order continuously differentiable function with $r \in \mathbb{Z}_{>0}$.

3.2 Threat Model

We consider that there exists an intelligent adversary in the CPS with the goal of driving the system into unsafe region of operations. The intelligent adversary can initiate cyber attacks by exploiting vulnerabilities in the cyber subsystem so as to intrude into the system. Once the adversary completes intrusion, it then gains access to software, actuator, and other peripherals. The adversary can then corrupt the actuator signal and arbitrarily manipulate the control input to be $\tilde{u}_{k\delta} \in \mathcal{U}$ for epoch $k \in \mathbb{Z}_{\geq 0}$. As a consequence, the system will deviate from the desired trajectories and may violate the safety constraint. Throughout this article, we assume that the adversary cannot physically access the components in CPS, e.g., physically damage the sensors and plant. We also assume that the system is not susceptible to external sensor spoofing or jamming attacks.

4 PROPOSED CYBER RESILIENT FRAMEWORK

In this section, we first introduce the timing behaviors of the CRAs, which will be incorporated in our framework. Then, we propose a hybrid system model to capture CPS employing different CRAs. Finally, we state the problem that we seek to solve in this article.

4.1 Timing Behaviors

In this subsection, we discuss the timing behaviors of the CPS employing different CRAs. We first identify a set of statuses of the cyber subsystem. We then discuss how the system evolves with time by specifying its timing behavior, which will help us to develop a hybrid system model for CPS that implements any of these architectures.

Based on the behavior of the cyber subsystem, we identify the *statuses* of the system as follows: $normal, corrupted, restart$, $restoration, safety\ controller\ driven(SC)$, $switching$ and $unsafe$. When the nominal controller is being used and the system is in the safety set C, the corresponding status is *normal*. In this status, the control input u follows the control policy $\mu(x)$. Status *corrupted* models the scenario where an adversary successfully intrudes into the system and corrupts the control input to $\tilde{u} \neq \mu(x)$. Status *restart* models the system when the controller is in the process of restart and thus the controller input will be zero during this status. When the CPS restores the compromised controller using the backup controller of BFT++ in order to recover from the attack, we model such status as *restoration*. At this status, the control input will not be accurate since the controller is not fully recovered from attack, and thus, it will deviate

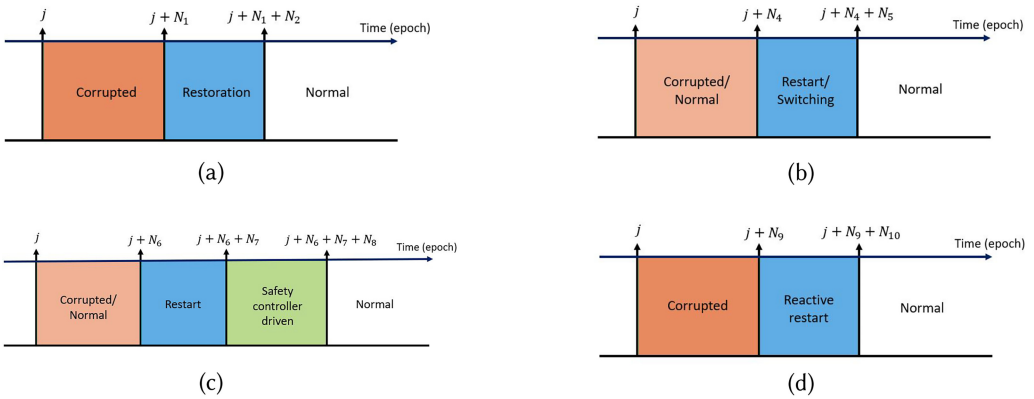

Fig. 1. This figure shows the timing behaviors of (a) BFT++, (b) YOLO/Dual redundant, (c) Proactive restart, and (d) Reactive restart architectures when an attacker corrupts the CPS at epoch index j by intruding into the cyber subsystem.

from the desired control signal. Status *SC* models the system when the safety controller drives the system. Switching process in dual redundant scheme is modeled as *switching*. The controller input will also be zero during *switching*. When the system is in the unsafe region, we model that as $unsafe$. We explicitly include $unsafe$ status in our model to capture the safety-critical nature of CPS. We remark that unlike other status in the hybrid model, the status $unsafe$ is solely based upon the physical states of the system, i.e., whether $x_t \notin C$.

Now we discuss the evolution of these statuses under attack. Suppose that an attacker intrudes the system at epoch index j, which causes the system status to change from *normal* to *corrupted*. The timing behavior of the system following the intrusion depends upon the employed architecture. We show the timing behaviors, i.e., evolutions of statuses of the CPS with time for different CRAs, in Figure 1.

When the system adopts BFT++ [27], the CPS will spend N_1 epochs in the *corrupted* status until one of the redundant controllers crashes due to diversity implemented in the controllers, e.g., diversified software implementation and randomized instruction set. This crash signal will initiate controller restoration. The system elapses N_2 epochs in *restoration* status during which the compromised controller are restored by using the backup controller. After restoration, the system will return to the *normal* status. In practice, for BFT++ implementation, we observe that $N_1 = 2$ and $N_2 = 2$ in the worst-case scenario [27]. Therefore if the physical subsystem can tolerate four epochs of disruption caused by a cyber attack, the system will remain safe. To withstand consecutive cyber attacks, the system needs to operate in the *normal* status for at least N_3 epochs.

In the case of YOLO [7, 8], it is possible for an attacker to exploit the vulnerabilities without causing any software crash. We suppose that the attacker can corrupt the system for at most N_4 epochs which is the up-time of the controller. During this *corrupted* status, the attacker manipulates the control signal issued by the controller. The controller will recover by the periodic restart triggered by a timer. The system will then elapse N_5 epochs in the *restart* status in order to reboot and reinitialize the controller. Note that at the *restart* status, the controller does not produce any control input and thus $u = 0$. The value of N_5 depends on multiple factors including operating system and controller, but in general $N_5 > N_2$. If the intrusion causes a software crash, then the system restarts automatically without a trigger coming from the timer. The restart period for YOLO is given by $N_4 + N_5$ epochs. The timing behavior of dual redundant scheme [15] is identical to YOLO except that the system periodically switches back to the other controller instead of periodic restart.

Proactive restart mechanism [1, 39] utilizes a timer to trigger the *restart* irrespective of whether the CPS is corrupted or not. Therefore, in the worst case, the system spends N_6 epochs in the *corrupted* status, which is same as the up-time of the main controller. The CPS elapses N_7 epochs in *restart* status during which a trusted image of software is loaded from read only memory unit. Unlike YOLO, a secured execution interval program, modeled as *SC*, is invoked for N_8 epochs after the restart. During this time, all the external interfaces are disabled, safety controller program is activated, and an external hardware timer is scheduled for next restart. After the end of this interval, the system returns to the *normal* status where the main controller program is activated and the external interfaces are enabled. Reference [1] uses online reachability analysis that determines the values of N_6 and N_8, whereas [39] provides an offline method to compute these timing parameters.

Unlike proactive restart, reactive restart mechanism does not have any timer for restart [32]. Instead, it relies upon crash signal as the trigger for restart. The system will remain in the *corrupted* status for N_9 epochs until crash causes the system to restart. During the *restart* status, the controller will be rebooted and reinitialized, which takes N_{10} epochs. Then, the CPS will resume the *normal* operation. It is assumed that there is an exploitation window of N_{11} epochs before which the adversary cannot intrude the system again [32].

The class of Simplex architectures [9, 29, 41] relies upon a verified controller which is assumed to be fault- and attack-free. A trusted decision module decides when the system switches to the safety controller by monitoring the system states or side channels. Decision module also decides when the main controller needs to be switched back from *SC* to *normal* where the main controller is active.

4.2 Proposed Framework

In what follows, we construct a hybrid system to model the CPS implementing the aforementioned CRAs. Our hybrid system is given by $H = (\mathcal{X}, \mathcal{U}, \mathcal{L}, \mathcal{Y}, \mathcal{Y}_0, Inv_x, Inv_u, \mathcal{F}, \Sigma, \Gamma)$ where

- $\mathcal{X} \subseteq \mathbb{R}^n$ is the continuous state space that represents the states of the physical subsystem.
- $\mathcal{U} \subseteq \mathbb{R}^m$ is the set of admissible control inputs of the physical subsystem.
- $\mathcal{L} = \{normal, corrupted, restart, restoration, SC, switching, unsafe\}$ is a set of discrete locations which corresponds to the statuses of the system. At each epoch, the system must be at some location $l \in \mathcal{L}$.
- $\mathcal{Y} = \mathcal{X} \times \mathcal{L}$ is the state space of the hybrid system H.
- $\mathcal{Y}_0$ is the set of initial states of the hybrid system H where $\mathcal{Y}_0 \subset \mathcal{Y}$.
- $Inv_x : \mathcal{L} \to 2^{\mathcal{X}}$ is the invariant that maps from the set of locations to the power set of $\mathcal{X}$. Function $Inv_x(l)$ specifies the set of possible continuous states when the system is at $l \in \mathcal{L}$.
- $Inv_u : \mathcal{L} \to 2^{\mathcal{U}}$ is the invariant that maps from the set of locations to the power set of $\mathcal{U}$. Function $Inv_u(l)$ specifies the set of admissible inputs when the system is at $l \in \mathcal{L}$.
- $\mathcal{F}$ is the set of vector fields. For each $F \in \mathcal{F}$, the continuous system state evolves as $\dot{x} = F(x, u, l)$, where $x \in Inv_x(l)$, $u \in Inv_u(l)$, and $\dot{x}$ is the time derivative of continuous state x. Here F is jointly determined by the system dynamics and the status of the cyber subsystem.
- $\Sigma \subseteq \mathcal{Y} \times \mathcal{Y}$ is the set of transitions between the states of the hybrid system H. A transition $\sigma = ((x, l), (x', l'))$ models the state transition from (x, l) to (x', l').
- Γ is a finite alphabet set. Each $\gamma \in \Gamma$ is labeled on some transition $\sigma \in \Sigma$ representing the events that triggers the transition.

Figure 2 presents our proposed hybrid model H. For convenience, we also show the valid components of the hybrid model for each CRA with other parts being removed in Figure 3. In these figures, each node (depicted by circles) represents a location $l \in \mathcal{L}$ and each directed edge (denoted by arrow) represents a transition $\sigma \in \Sigma$. We remark that in our setting, there is no discontinuity

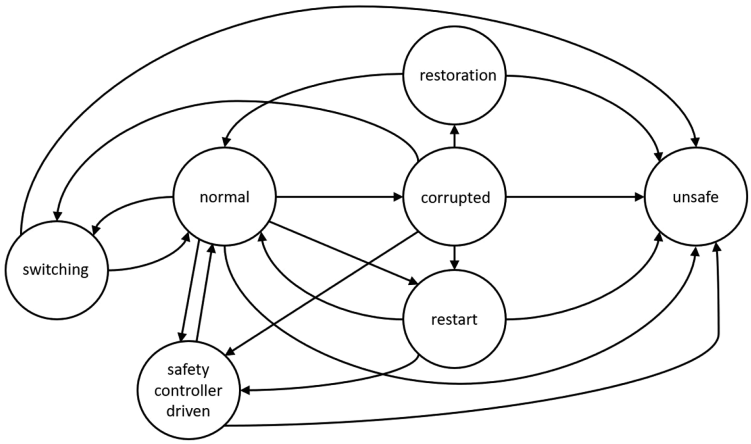

Fig. 2. This figure illustrates the proposed hybrid system model H. The statuses are depicted as circles, and transitions among the statuses are captured by arrows. This hybrid model captures the evolution of cyber statuses of the CPS that employ resilient architectures.

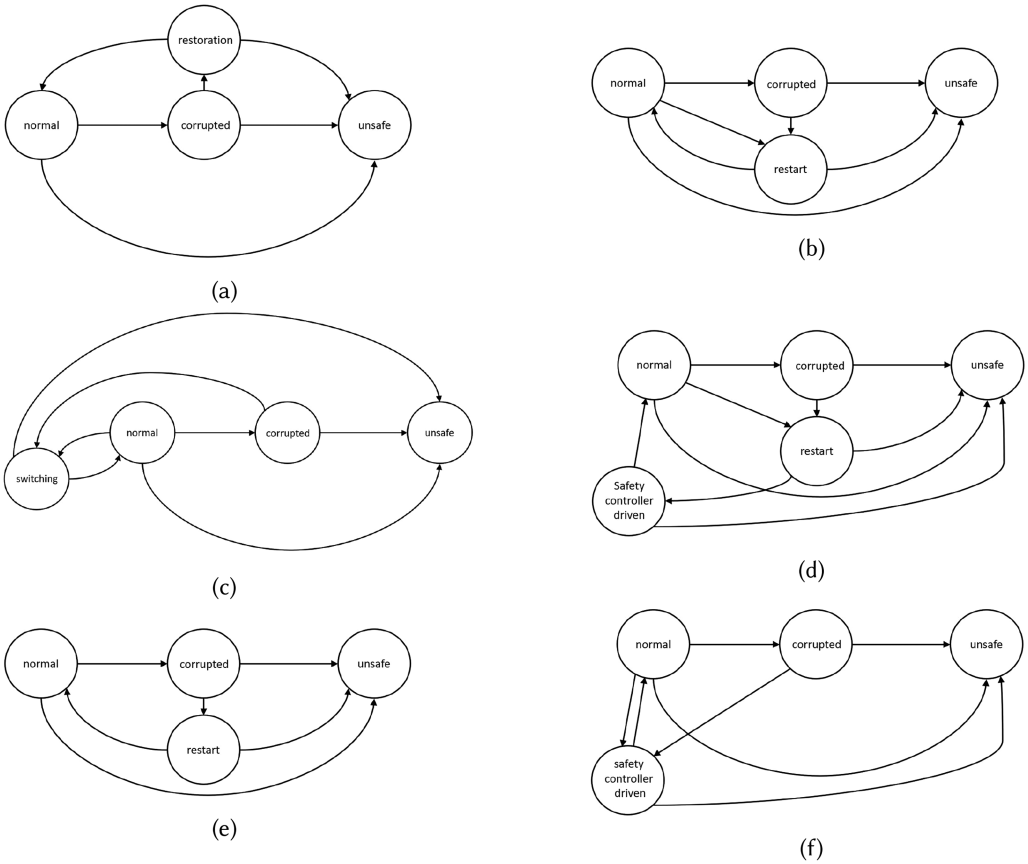

Fig. 3. This figure shows the components (a) BFT++, (b) YOLO, (c) Dual redundant, (d) Proactive restart, (e) Reactive restart, and (f) Simplex/S3A architectures in the proposed hybrid system model H.

Table 1. This Table Lists the Transitions in the Hybrid System Model H

Transition	Label	Valid Architecture(s)
$(normal, corrupted)$	*cyber intrusion*	Any CRA [1, 7–9, 15, 27, 29, 32, 39, 41]
$(corrupted, restoration)$	*controller crash*	BFT++ [27]
$(restoration, normal)$	*controller restored*	BFT++ [27]
$(corrupted, restart)$	*controller crash*	YOLO [7, 8], Reactive Restart [32]
$(corrupted, restart)$	*restart timer*	YOLO [7, 8], Proactive Restart [1, 39]
$(normal, restart)$	*restart timer*	YOLO [7, 8], Proactive Restart [1, 39]
$(restart, normal)$	*controller restarted*	YOLO [7, 8], Reactive Restart [32]
$(restart, SC)$	*controller restarted*	Proactive Restart [1, 39]
$(SC, normal)$	*SEI ended*	Proactive Restart [1, 39]
$(normal, switching)$	*switching timer*	Dual Redundant [15]
$(corrupted, switching)$	*switching timer*	Dual Redundant [15]
$(switching, normal)$	*switching completed*	Dual Redundant [15]
$(corrupted, SC)$	*safety controller invoked*	Simplex/S3A [9, 29, 41]
$(normal, SC)$	*safety controller invoked*	Simplex/S3A [9, 29, 41]
$(SC, normal)$	*main controller invoked*	Simplex/S3A [9, 29, 41]
$(l, unsafe)_{l \neq unsafe}$	*safety region crossed*	Any CRA [1, 7–9, 15, 27, 29, 32, 39, 41]

For each transition in the first column, the corresponding label and the corresponding architectures for which the transition is valid are listed in the second and third column respectively.

in the physical states and epoch indices for any of the transitions. Hence for simplicity we will denote a transition $\sigma = ((x, l), (x', l')) \in \Sigma$ as (l, l'). Note that some transitions are valid for specific architectures employed by the CPS. For example, transition $(corrupted, restoration)$ is valid for BFT++ while the transition from $(corrupted, restart)$ is not. A run (i.e., a sequence of successive transitions) in the model H determines the evolution of the CPS statuses with time. Tracking successive transitions in the model H provides a means to analyze CPS safety.

Each transition is labeled with an alphabet set $\gamma \in \Gamma$ where γ represents the event that triggers the transition. For example, the transition $(normal, corrupted)$ is triggered by a cyber intrusion whereas the transition $(corrupted, restoration)$ is triggered by a controller crash. The details of all labels and corresponding transitions are given in Table 1.

In Figure 2, we observe that location $l = unsafe$ is absorbing since it has an incoming edge from each other node and does not have any outgoing edge. It implies that if the CPS ever transits to $l = unsafe$, it will remain there since the safety constraint has been violated. Thus, our focus in this study is to avoid any run on the hybrid model H that reaches $l = unsafe$.

4.3 Problem Formulation

This subsection states the problem of interest. We consider the hybrid system H that represents the CPS employing any of the CRAs. Let t_1 be the time instant when the system transits to *corrupted* status triggered by a cyber intrusion. The CRAs employed by the system tries to recover the system so that the system goes back to *normal* status at some time instant $\tilde{t} > t_1$. Let t_2 be the first time when the system again transits to *corrupted* status following a new cyber intrusion where $t_2 \geq \tilde{t}$. We define the time interval $[t_1, t_2]$ as an *attack cycle*. The length $A = t_2 - t_1$ of each attack cycle varies and is dependent on the capabilities of the adversary and the vulnerability of cyber subsystem. Here, we focus on guaranteeing the system safety for an

arbitrary attack cycle and thus for all time $t \geq 0$. Our problem in this article is formalized as follows:

Problem 4.1. *Synthesize a control policy $\mu : \mathcal{X} \times \{normal, SC\} \rightarrow \mathcal{U}$ and design the timing parameters for the CPS such that the system never reaches the location $l = unsafe$ during an attack cycle.*

Here, the timing parameters quantify how many epochs the system can remain at each status. The above problem is equivalent to computing a control policy and the timing parameters such that $x_t \in C \; \forall [t_1, t_2]$. Our method for computation of the control policy and timing parameters is detailed in Section 5.

5 PROPOSED SOLUTION APPROACH

In this section, we first provide some preliminary background on barrier certificates and SOS programming. Then, we present our proposed solution to Problem 4.1.

5.1 Preliminaries

In this subsection, we present necessary preliminaries which will be useful in deriving our solution to the formulated problem. A continuous function $\alpha : [-b, a) \rightarrow (-\infty, \infty)$ belongs to the extended class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$ for some $a, b > 0$. A set C is called forward invariant if $x_t \in C \; \forall t \geq t_0$ given that $x_{t_0} \in C$.

The Lie derivative of function $h(x)$ with respect to dynamics Equation (1) is given by $L_f h(x) + L_g h(x)u$ where $L_f h(x) = \frac{\partial h}{\partial x}(x) f(x)$ and $L_g h(x) = \frac{\partial h}{\partial x}(x) g(x)$ denote the Lie derivatives along $f(x)$ and $g(x)$, respectively. The higher order Lie derivatives of $h(x)$ along $f(x)$ are obtained inductively via $L_f^i h(x) = L_f L_f^{i-1} h(x)$ where $i \in \mathbb{Z}_{>0}$ and $L_f^0 h(x) = h(x)$. *Relative degree* of a continuously differentiable function on a set with respect to a dynamics is defined by the minimum number of times Lie derivatives of the function needs to be taken along the dynamics such that control input explicitly appears in the expression [21, 43]. This is formally defined as follows:

Definition 5.1 ([21, 43]). The relative degree of a function $h(x)$ is $r \in \mathbb{Z}_{>0}$ on the set $\mathcal{X}$ with respect to dynamics Equation (1) if $h(x)$ is rth order continuously differentiable on $\mathcal{X}$ and $L_g L_f^{r-1} h(x) \neq 0$ and $L_g L_f^i h(x) = 0$ for all $i \in \{0, 1, \ldots, r-2\}$ and for all $x \in \mathcal{X}$.

In this aticle, we denote the ith order Lie derivative of $h(x)$ as $h^i(x)$. We abuse this notation by assuming $h^i(x) = h(x)$ when $i = 0$. Note that we can write $h^i(x) = L_f^i h(x), \; \forall i = \{0, 1, \ldots r-1\}$ and $h^r(x) = L_f^r h(x) + L_g L_f^{r-1} h(x) \mu(x)$ given that the relative degree of $h(x)$ is r and u follows some control policy $\mu(x)$.

A multivariate polynomial $p(x)$ is a SOS polynomial if there exists a set of polynomials $q_1(x), q_2(x), \ldots, q_l(x)$ such that $p(x) = \sum_{i=1}^{l} (q_i(x))^2$. If $p(x)$ is a SOS polynomial then we have that $p(x) \geq 0$. SOS and non-negativity are equivalent when the polynomial is quadratic or univariate [35]. In this article, we denote the set of SOS polynomials over $x \in \mathbb{R}^n$ as $\mathcal{S}(x)$.

SOS program is an established method for solving optimization problems with polynomial constraints and objective functions. SOS program converts the original problem to a **semidefinite program** (**SDP**), which can be solved using convex optimization. However, as the size of the polynomial increases, the converted SDP becomes huge and increasingly ill-conditioned—which may cause numerical instability of SOS programs. Some techniques [3] have been developed to partially alleviate this problem. We remark that we did not encounter such numerical instability issues in our case study.

5.2 Control Synthesis and Timing Parameters Design for Safety

In this subsection, we present our proposed solution approach to control synthesis and timing parameters design for the system's safety. We first derive a set of sufficient conditions for a control policy along with timing parameters so that safety of the CPS is guaranteed. Next, we encode the derived conditions as a set of SOS constraints under certain assumptions. Using the SOS formulation, we propose an algorithm to compute the control policy and associated timing parameters. We also analyze convergence of our proposed algorithm.

Recall that safety of the CPS is defined over a set $C = \{x \in \mathcal{X} : h(x) \geq 0\}$. We suppose that the relative degree of the function $h(x)$ on set C with respect to dynamics Equation (1) is r. When $r > 1$, the control input u does not appear in the first order Lie derivative of $h(x)$ since $L_g h(x) = 0$. In that case, solution methods based on first order Lie derivative of $h(x)$ [31, 32] are not applicable to guarantee safety anymore. To address this limitation, here, we derive sufficient conditions for control policy which holds for any $r \in \mathbb{Z}_{>0}$. To do so, we first derive a result on the forward invariance of a function for higher order Lie derivatives. We present the result below.

Lemma 5.2. *Let $\mathcal{A} = \{x_t : h(x_t) \geq c_0\} \cap \{x : h^1(x_t) \geq c_1\} \cap \cdots \cap \{x_t : h^{p-1}(x_t) \geq c_{p-1}\}$ for some $c_0, c_1, \ldots, c_{p-1} \geq 0$ and $p = 1, 2, \ldots, r$ where r is the relative degree of $h(x)$ on the set C with respect to dynamics (1). If $h^p(x_t) + \alpha(h^{p-1}(x_t) - c_{p-1}) \geq 0$ holds for all $x_t \in \mathcal{A}$, then $\mathcal{A}$ is forward invariant. That is, $x_t \in \mathcal{A}$ for all $t \geq t_0$ given that $x_{t_0} \in \mathcal{A}$.*

Proof. We denote the boundary of a set $\mathcal{A}_i = \{x : h^i(x_t) \geq c_i\}$ as

$$\partial\mathcal{A}_i = \{x : h^i(x_t) = c_i\},\ \forall i = 0, 1, \ldots, p-1.$$

Suppose $x_{t_0} \in \mathcal{A}$. Note that when $x_t \in \partial\mathcal{A}_{p-1} \cap \mathcal{A}$, from $h^p(x_t) + \alpha(h^{p-1}(x_t) - c_{p-1}) \geq 0$ we get $h^p(x_t) \geq -\alpha(0) = 0$. This implies $h^{p-1}(x_t) \geq c_{p-1}\ \forall t \geq t_0$. This also completes the proof for the case $p = 1$. Now we consider the case $p \geq 2$. For this case, we first show that $h^i(x_t) \geq c_i,\ \forall t \geq t_0,\ \forall i = 0, 1, \ldots, p-2$. Since $c_0, c_1, \ldots, c_{p-1} \geq 0$, we have that $h^i(x_t) \geq 0\ \forall i = 0, 1, \ldots, p-1$ if $x_t \in \mathcal{A}$. Now consider the case where $x_t \in \partial\mathcal{A}_i \cap \mathcal{A}$ for any $i = 0, 1, \ldots, p-2$. Since we have $h^{i+1}(x_t) \geq 0$, according to Nagumo's theorem [10], we have that $h^i(x_t) \geq c_i,\ \forall t \geq t_0$. Since this holds for any $i = 0, 1, \ldots, p-2$ and previously we have shown that $h^{p-1}(x_t) \geq c_{p-1},\ \forall t \geq t_0$, hence we conclude that $x_t \in \mathcal{A}$ for all $t \geq t_0$. This completes the proof for $p = 1, 2, \ldots, r$. □

Now using the above result, we will derive sufficient conditions for a control policy to ensure safety in the presence of cyber attack, considering that the policy is implemented as specified by the timing parameters. We will derive the conditions which can be applied to any CPS regardless of its resilient architecture. We will consider an arbitrary run on the hybrid system H that can possibly visit any number of locations before returning to *normal* for the purpose of generality. This run starts from the *corrupted* location following a cyber attack. Our objective is to find a level set $\mathcal{A}$ and control policy $\mu(x)$ for the recovered or safety controller so that the CPS remains safe (i.e., never reach the location $l = unsafe$) during any attack cycle. We also enforce that the CPS returns to the computed level set $\mathcal{A}$ so that safety of the CPS can be guaranteed for consecutive attacks. Our derived conditions are presented in the following theorem.

Theorem 5.3. *Consider the hybrid model H from Section 4. Let $\mathcal{A}$ be a level set defined over higher order Lie derivatives of $h(x)$ for some $c_0, c_1, \ldots, c_{r-1} \geq 0$ as*

$$\mathcal{A} = \{x : h(x) \geq c_0\} \cap \{x : L_f h(x) \geq c_1\} \cap \cdots \cap \left\{x : L_f^{r-1} h(x) \geq c_{r-1}\right\}.$$

Consider an attack cycle $[t_1, t_2]$ for which the system undergoes a sequence of successive transitions $(l_1, l_2), (l_2, l_3), \ldots, (l_{k-1}, l_k)$ in the hybrid model until the controllers are recovered or safety controller is invoked, where $l_1 = corrupted$. Let $\tau_1, \tau_2, \ldots, \tau_k + \Delta$ be the amount of time elapsed in each locations

$l_1, l_2, \ldots, l_k$, respectively, where $\Delta = (t_2 - t_1) - \sum_{i=1}^{k} \tau_i \geq 0$. If there exist constants $s_1, s_2, \ldots, s_k$ and a control policy $\mu(x) \in Inv_u(l_k)$ which is applied at location l_k such that

$$L_f^r h(x) + L_g L_f^{r-1} h(x) u - s_j \geq 0,\ \forall (x,u) \in C \times Inv_u(l_j), \forall j = 1, 2, \ldots, k-1 \tag{2a}$$

$$L_f^r h(x) + L_g L_f^{r-1} h(x) \mu(x) - s_k \geq 0,\ \forall x \in C \backslash \mathcal{A} \tag{2b}$$

$$L_f^r h(x) + L_g L_f^{r-1} h(x) \mu(x) + \alpha\left(L_f^{r-1} h(x) - a_{0,r-1}\right) \geq 0,\ \forall x \in \mathcal{A} \tag{2c}$$

$$\sum_{i=1}^{r} \frac{a_{j-1,r-i}(t-\xi_j)^{r-i}}{(r-i)!} + \frac{s_j (t-\xi_j)^r}{r!} \geq 0,\ \forall t \in [\xi_j, \xi_{j+1}], \forall j = 1, 2, \ldots, k \tag{2d}$$

$$a_{k,r-p} \geq a_{0,r-p},\ \forall p = 1, 2, \ldots, r \tag{2e}$$

where

$$a_{0,i} = c_i, \forall i = 0, 1, \ldots, r-1 \tag{3a}$$

$$a_{j,r-p} = \sum_{i=1}^{p} \frac{a_{j-1,r-i} \tau_j^{p-i}}{(p-i)!} + \frac{s_j \tau_j^p}{p!},\ \forall p = 1, 2, \ldots, r;\ \forall j = 1, 2, \ldots, k \tag{3b}$$

$$\xi_j = \sum_{i=0}^{j-1} \tau_i,\ \forall j = 1, 2, \ldots, k+1 \text{ with } \tau_0 = t_1. \tag{3c}$$

then the system Equation (1) is safe i.e., $x_t \in C,\ \forall t \in [t_1, t_2]$ given that $x_{t_1} \in \mathcal{A}$. Furthermore, $x_t \in \mathcal{A}$ for $t \in [\xi_{k+1}, t_2]$.

Proof. From the statement of the theorem, we suppose that $x_{t_1} \in \mathcal{A}$, which implies

$$h(x_{t_1}) \geq c_0 = a_{0,0}, \quad h^1(x_{t_1}) \geq c_1 = a_{0,1}, \ldots,\ h^{r-1}(x_{t_1}) \geq c_{r-1} = a_{0,r-1}.$$

We note that $a_{j,p}$ corresponds to a lower bound on the value of $h^p(x)$ when the system transitions to the location l_{j+1} and ξ_{j+1} corresponds to the time of that transition. First, consider the case where the system is at location l_1. For this case, we will show that $h^p(x_{\xi_2}) \geq a_{1,p}$ for $p = 0, 1, \ldots, r-1$ and $x_t \in C$ for $t \in [t_1, t_1 + \tau_1] = [\xi_1, \xi_2]$. We will show this by integrating over $h^r(x)$ inductively r times with initial condition at time $t = \tau_0 = \xi_1$ and using the fact that $h^r(x) = L_f^r h(x) + L_g L_f^{r-1} h(x) u$. Integrating over $h^r(x_t)$ first and using Equation (2a) with $j = 1$, we can write for all $(x,u) \in C \times Inv_u(l_1)$ and $t \in [\xi_1, \xi_2]$

$$h^{r-1}(x_t) = h^{r-1}(x_{\xi_1}) + \int_{t=\xi_1}^{t} h^r(x_t)\, dt \geq a_{0,r-1} + s_1(t - \xi_1). \tag{4}$$

By integrating over $h^{r-1}(x_t)$ next, we have that for all $(x,u) \in C \times Inv_u(l_1)$ and $t \in [\xi_1, \xi_2]$

$$\begin{aligned} h^{r-2}(x_t) &= h^{r-2}(x_{\xi_1}) + \int_{t=\xi_1}^{t} h^{r-1}(x_t)\, dt \\ &\geq a_{0,r-2} + \int_{t=\xi_1}^{t} (a_{0,r-1} + s_1(t-\xi_1))\, dt = a_{0,r-2} + a_{0,r-1}(t-\xi_1) + \frac{s_1(t-\xi_1)^2}{2}, \end{aligned} \tag{5}$$

where the inequality follows from Equation (4) and the fact that $h^{r-2}(x_{\xi_1}) \geq c_{r-2} = a_{0,r-2}$. Thus if we continue this by inductively integrating over $h^r(x_t)$ for p times where $p = 1, 2, \ldots, r$, we have that for all $(x,u) \in C \times Inv_u(l_1)$ and $t \in [\xi_1, \xi_2]$

$$h^{r-p}(x_t) \geq \sum_{i=1}^{p} \frac{a_{0,r-i}(t-\xi_1)^{p-i}}{(p-i)!} + \frac{s_1(t-\xi_1)^p}{p!}, \tag{6}$$

Note from Equation (3c), we have $\xi_2 = \tau_0 + \tau_1 = \xi_1 + \tau_1$. Now Substituting $t = \xi_2$ in Equation (6) and using Equation (3b) with $j = 1$ we have that for all $p = 1, 2, \ldots, r$

$$h^{r-p}(x_{\xi_2}) \geq \sum_{i=1}^{p} \frac{a_{0,r-i}(\xi_2 - \xi_1)^{p-i}}{(p-i)!} + \frac{s_1(\xi_2 - \xi_1)^p}{p!} = \sum_{i=1}^{p} \frac{a_{0,r-i}\tau_1{}^{p-i}}{(p-i)!} + \frac{s_1\tau_1{}^p}{p!} = a_{1,r-p}. \tag{7}$$

Furthermore, by letting $r = p$ in Equation (6) and using Equation (2d) with $j = 1$, we get for all $t \in [\xi_1, \xi_2]$

$$h(x_t) \geq \sum_{i=1}^{r} \frac{a_{0,r-i}(t - \xi_1)^{r-i}}{(r-i)!} + \frac{s_1(t - \xi_1)^r}{r!} \geq 0. \tag{8}$$

Therefore, $x(t) \in C,\ \forall t \in [\xi_1, \xi_2]$. Now consider the case where the system is at location l_2. For this case we will similarly show that $h^p(x_{\xi_3}) \geq a_{2,p}$ for $p = 0, 1, \ldots, r-1$ and $x_t \in C$ for $t \in [\xi_2, \xi_3]$. Integrating over $h^r(x_t)$ and using Equation (2a) with $j = 2$, we can write for all $(x, u) \in C \times Inv_u(l_2)$ and $t \in [\xi_2, \xi_3]$

$$h^{r-1}(x_t) = h^{r-1}(x_{\xi_2}) + \int_{t=\xi_2}^{t} h^r(x_t)\, dt \geq a_{1,r-1} + s_2(t - \xi_2). \tag{9}$$

The last inequality follows from $h^{r-1}(x_t) \geq a_{1,r-1}$ which we obtain from Equation (7) by letting $p = 1$. Similar to before, by repeated integration, we can write for all $(x, u) \in C \times Inv_u(l_2)$, $t \in [\xi_2, \xi_3]$ and $p = 1, 2, \ldots, r$

$$h^{r-p}(x_t) \geq \sum_{i=1}^{p} \frac{a_{1,r-i}(t - \xi_2)^{p-i}}{(p-i)!} + \frac{s_2(t - \xi_2)^p}{p!}. \tag{10}$$

Like before, substituting $t = \xi_3$ in Equation (10) and using Equations (3b) and (3c), we have that $h^{r-p}(x_{\xi_3}) \geq a_{2,r-p}$ for all $p = 1, \ldots, r$. Also by letting $r = p$ in Equation (10) and using Equation (2d) with $j = 2$, we get $h(x_t) \geq 0$ for all $t \in [\xi_2, \xi_3]$.

If we continue the above steps for locations $l_3, \ldots, l_{k-1}$ we can show that for all $(x, u) \in C \times Inv_u(l_j)$ and $t \in [\xi_j, \xi_{j+1}]$ where $j = 1, 2, \ldots, k-1$ and $p = 1, 2, \ldots, r$

$$h^{r-p}(x_t) \geq \sum_{i=1}^{p} \frac{a_{j-1,r-i}(t - \xi_j)^{p-i}}{(p-i)!} + \frac{s_j(t - \xi_j)^p}{p!}, \tag{11}$$

$h^{r-p}(x_{\xi_{j+1}}) \geq a_{j,r-p}$ and $h(x_t) \geq 0$ i.e., $x(t) \in C$.

Now consider the last case when the system is at location l_k where control policy $\mu(x)$ is applied. Integrating Equation (2b) for p times and using the fact $h^{r-p}(x_{\xi_{j+1}}) \geq a_{j,r-p}$ where $j = 1, 2, \ldots, k-1$ and $p = 1, 2, \ldots, r$, we can write for all $x \in C \backslash \mathcal{A}$ and $t \in [\xi_k, \xi_{k+1}] \cup [\xi_{k+1}, \xi_{k+1} + \Delta] = [\xi_k, t_2]$

$$h^{r-p}(x_t) \geq \sum_{i=1}^{p} \frac{a_{k-1,r-i}(t - \xi_k)^{p-i}}{(p-i)!} + \frac{s_k(t - \xi_k)^p}{p!}. \tag{12}$$

Now Equations (12) and (2d) with $j = k$ implies that $h(x_t) \geq 0$, i.e., $x(t) \in C,\ \forall t \in [\xi_k, \xi_{k+1}]$. Furthermore, substituting $t = \xi_{k+1}$ in Equation (12) and using Equations (3b) and (3c), we get that $h^{r-p}(x_{\xi_{k+1}}) \geq a_{k,r-p}\ \forall p = 1, 2, \ldots, r$. This along with Equation (2e) imply that for all $p = 1, \ldots, r$

$$h^{r-p}(x_{\xi_{k+1}}) \geq a_{k,r-p} \geq a_{0,r-p} = c_{r-p}.$$

Therefore, $x(\xi_{k+1}) \in \mathcal{A}$. Now, we show that $x_t \in \mathcal{A}$ for the remaining time of the attack cycle i.e., $\forall t \in [\xi_{k+1}, t_2] = [t_2 - \Delta, t_2]$. From Equation (2c) and $x(\xi_{k+1}) \in \mathcal{A}$, we note that we can apply Lemma 5.2 for $p = r$. Lemma 5.2 implies that $x_t \in \mathcal{A},\ \forall t \in [\xi_{k+1}, t_2] = [t_2 - \Delta, t_2]$. Since $\mathcal{A} \subset C$ and $\cup_{i=1}^{k}[\xi_i, \xi_{i+1}] \cup [\xi_{k+1}, t_2] = [t_1, t_2]$, thereby, we have $x_t \in C\ \forall [t_1, t_2]$. This completes the proof. □

The above theorem gives a set of sufficient conditions for control policy and timing parameters of any employed CRA so that safety of the CPS is guaranteed. The value of k, the locations $l_1, l_2, \ldots, l_k$ and the timing parameters $\tau_1, \tau_2, \ldots, \tau_k$ depend on the architecture adopted by the CPS. For example, for BFT++ we have $k = 3$, $l_1 = corrupted$, $l_2 = restoration$, $l_3 = normal$, $\tau_1 = N_1$ epochs, $\tau_2 = N_2$ epochs and $\tau_3 = N_3$ epochs. Therefore at $l = normal$ location the restored controller needs to apply the control policy $\mu(x)$ for at least τ_3 time to guarantee safety. As the system returns to the level set $\mathcal{A}$ within one attack cycle, safety is also ensured for consecutive attacks as long as the attack cycle is greater than $\tau_1 + \tau_2 + \tau_3$. The mapping between the arbitrary parameters $\tau_1, \tau_2, \ldots, \tau_k$ in Theorem 5.3 and the timing parameters of CRAs is discussed in detail later in Table 2.

Conditions given in Theorem 5.3 are more general than those derived in [31, 32] as it applies to the cases where relative degree $r > 1$ and number of locations $k > 3$. We can derive simpler form of Theorem 5.3 by using specific values of r and k. For example, by letting $r = 1$ and $k = 2$ we can recover the result in [31] for hard safety constraint. This simplified form can be used for the worst-case analysis of a CPS with $r = 1$ by considering that control input at any location $l \in \{corrupted, restoration, restart, switching\}$ are malicious. Such result is given below.

Corollary 5.4. *For the hybrid system model H from Section 4, let $\mathcal{A}$ be a level set defined as $\mathcal{A} = \{x : h(x) \geq c\}$. Consider an attack cycle $[t_1, t_2]$ where control input is malicious for η time until the controller recovers. If there exist constants $c \geq 0$ and a control policy $\mu : \mathcal{X} \to \mathcal{U}$ such that*

$$\frac{\partial h}{\partial x}(x)(f(x) + g(x)u) + \frac{c}{\eta} \geq 0, \ \forall (x, u) \in C \times \mathcal{U} \tag{13a}$$

$$\frac{\partial h}{\partial x}(x)(f(x) + g(x)\mu(x)) - \frac{c}{\tau} \geq 0, \ \forall x \in C \setminus \mathcal{A} \tag{13b}$$

$$\frac{\partial h}{\partial x}(x)(f(x) + g(x)\mu(x)) + \alpha(h(x) - c) \geq 0, \ \forall x \in \mathcal{A} \tag{13c}$$

then the system Equation (1) is safe i.e., $x_t \in C$, $\forall t \in [t_1, t_2]$ by taking policy μ as desired control input given that $x_{t_1} \in \mathcal{A}$ and $\eta + \tau \leq t_2 - t_1$. Furthermore, $x_t \in \mathcal{A}$ for all $t \in [t_1 + \eta + \tau, t_2]$.

Proof. We can prove the corollary by substituting $r = 1$, $k = 2$, $l_1 = corrupted$, $l_2 = normal$, $c_0 = c, \tau_1 = \eta, \tau_2 = \tau, s_1 = -\frac{c}{\eta}$ and $s_2 = \frac{c}{\tau}$ into Theorem 5.3. In this case we note that Equations (13a), (13b), and (13c) are directly obtained from Equations (2a), (2b) and (2c). Condition Equation (2d) becomes $c - \frac{c}{\eta}(t - t_1) \geq 0$, $\forall t \in [\xi_1, \xi_2] = [t_1, t_1 + \eta]$ for $j = 1$. This condition is trivial since $0 \leq \frac{t - t_1}{\eta} \leq 1$ as $t \in [t_1, t_1 + \eta]$. Note from Equation (3b), we have $a_{1,0} = c - (\frac{c}{\eta})\eta = 0$. Therefore, for $j = 2$, condition Equation (2d) is again trivially satisfied since $a_{1,0} = 0$ and $s_2 = \frac{c}{\tau} \geq 0$. Note from Equation (3b), we have $a_{2,0} = 0 + (\frac{c}{\tau})\tau = c$. Hence, the condition Equation (2e) becomes trivial as it yields $c \geq c$. Thus conditions Equation (13) are sufficient which completes the proof. □

The conditions in Theorem 5.3 and Corollary 5.4 focus on stringent safety constraints. We remark that our approach can be readily extended to derive sufficient conditions for control policies under which the CPS is allowed to operate outside the safety set C at the expense of incurring cost [31]. The cost is given by a non-decreasing function J such that $J(h(x_t)) \geq 0$ if $h(x_t) \leq 0$ and zero otherwise. In this case, the goal is to synthesize a control policy that satisfies a budget constraint B on the total incurred cost $\int_{t=t_1}^{t_2} J(h(x_t)\, dt \leq B$. We can achieve this by omitting the condition Equation (2d) in Theorem 5.3 and replacing C with $\mathcal{D} = \{x \in \mathcal{X} : h(x) \geq -d\}$ for some $d \geq 0$ in the conditions Equations (2a) and (2b). We then compute the upper bound on incurred cost by

utilizing the fact that

$$h(x_t) \geq \sum_{i=1}^{r} \frac{a_{j-1,r-i}(t-\xi_j)^{r-i}}{(r-i)!} + \frac{s_j(t-\xi_j)^r}{r!}$$

for all $j = 1, \ldots, k$ and for all $t \in [\xi_1, \xi_{k+1}]$. This bound can be encoded as a constraint given as

$$\sum_{j=1}^{k} \int_{t=\xi_j}^{\xi_{j+1}} J\left(\sum_{i=1}^{r} \frac{a_{j-1,r-i}(t-\xi_j)^{r-i}}{(r-i)!} + \frac{s_j(t-\xi_j)^r}{r!}\right) dt \leq B$$

when synthesizing the control policy. Since for safety-critical CPS the safety requirements are stringent, we omit consideration of this soft constraint case in our development.

In the following, we investigate how to synthesize the control policy μ and other corresponding parameters, e.g., $c_1, \ldots, c_r$, $\tau_1, \ldots, \tau_k$ to guarantee safety of the CPS. Our approach is to translate the inequality conditions given in Theorem 5.3 into SOS constraints and then solve the resulting SOS program. By noting that many real-world safety-critical systems can be represented (either exact or approximate sense) using polynomial dynamics [6, 26], we make the following assumption for our purpose.

Assumption 5.1. *We assume that functions $f(x)$, $g(x)$, and $h(x)$ are polynomial in x.*

The assumption above allows us to derive SOS formulation of Theorem 5.3. The SOS formulation is given by the following result.

Proposition 5.5. *Suppose there exist parameters $c_0, c_1, \ldots, c_{r-1} \geq 0$; $\tau_1, \tau_2, \ldots, \tau_k \geq 0$, and $s_1, s_2 \ldots, s_k$ such that*

$$L_f^r h(x) + L_g L_f^{r-1} h(x) u - s_j - q_j(x,u) h(x) - \sum_{i=1}^{m} \left(w_{j,i}(x,u)([u]_i - [u]_{l_j,i}^{min}) + v_{j,i}([u]_{l_j,i}^{max} - [u]_i) \right) \in \mathcal{S}(x,u), \forall j = 1, 2, \ldots, k-1 \tag{14a}$$

$$L_f^r h(x) + L_g L_f^{r-1} h(x) \lambda(x) - s_k + \sum_{i=0}^{r-1} p_i(x) \left(L_f^i h(x) - a_{0,i} \right) - l(x) h(x) \in \mathcal{S}(x) \tag{14b}$$

$$L_f^r h(x) + L_g L_f^{r-1} h(x) \lambda(x) + \alpha\left(L_f^{r-1}(x) - a_{0,r-1} \right) - \sum_{i=0}^{r-1} z_i(x) \left(L_f^i h(x) - a_{0,i} \right) \in \mathcal{S}(x) \tag{14c}$$

$$\lambda_i(x) - [u]_{l_k,i}^{min} \in \mathcal{S}(x), \quad [u]_{l_k,i}^{max} - \lambda_i(x) \in \mathcal{S}(x), \ \forall i = 1, 2, \ldots, m, \tag{14d}$$

$$\sum_{i=1}^{r} \frac{a_{j-1,r-i}(t-\xi_j)^{r-i}}{(r-i)!} + \frac{s_j(t-\xi_j)^r}{r!} - \phi_j(t)(t-\xi_j) + \psi_j(t)(t-\xi_{j+1}) \in \mathcal{S}(t), \ \forall j = 1, 2, \ldots, k \tag{14e}$$

and the following inequality holds:

$$a_{k,r-p} \geq a_{0,r-p}, \ \forall p = 1, 2, \ldots, r, \tag{15}$$

where $a_{j,i}$ $\forall j = 0, 1, \ldots, k$ $\forall i = 0, 1, \ldots, r-1$, and ξ_j $\forall j = 0, 1, \ldots, k+1$ are given by (3) and $l(x), p_i(x), z_i(x) \in \mathcal{S}(x)$ $\forall i = 0, 1, \ldots, r-1$; $\lambda_i(x)$ is a polynomial in x for each $i = 1, \ldots, m$; $q_j(x,u), w_{j,i}(x,u), v_{j,i}(x,u) \in \mathcal{S}(x,u)$, $\forall i = 1, 2, \ldots, m$ $\forall j = 1, 2, \ldots, k-1$ and $\phi_j(t), \psi_j(t) \in \mathcal{S}(t)$, $\forall j = 1, 2, \ldots, k$. Then the control policy $\mu(x) = \lambda(x) = [\lambda_1(x), \ldots, \lambda_m(x)]^\top$ satisfies the conditions in (2) for the parameters $c_0, c_1, \ldots, c_{r-1}$; $\tau_1, \tau_2, \ldots, \tau_k$ and $s_1, s_2 \ldots, s_k$.

Proof. First, we will show that Equation (14a) implies Equation (2a). Consider $x \in C$ and $u \in Inv_u(l_j)$ which implies $[u]_{l_j,i}^{min} \leq [u]_i \leq [u]_{l_j,i}^{max}, \forall i = 1, \ldots, m$ where $j = 1, 2, \ldots, k-1$. Therefore

ALGORITHM 1: Heuristic Algorithm for Computing $c_0, \ldots, c_{r-1}$; $\tau_1, \ldots, \tau_k$ and Control Policy $\mu(x)$

Input: $f(x); g(x); h(x); c_i^{max}\ \forall i = 0, 1, \ldots, r-1$; τ_j^{min} and $\tau_j^{max}\ \forall j = 1, 2, \ldots, k$
Output: $c_i\ \forall i = 0, 1, \ldots, r-1$; $\tau_j\ \forall j = 1, 2, \ldots, k$ and $\mu(x)$
Initialization: $c_i = 0\ \forall i = 0, 1, \ldots, r-1$
loop sweep $(c_0, c_1, \ldots, c_{r-1})$ from $(0, 0, \ldots, 0)$ to $(c_0^{max}, c_1^{max}, \ldots, c_{r-1}^{max})$
    Maximize s_j subject to (14a) $\forall j = 1, \ldots, k-1$ and maximize s_k subject to (14b), (14c), (14d).
    loop sweep $(\tau_1, \tau_2, \ldots, \tau_k)$ from $(\tau_1^{max}, \tau_2^{max}, \ldots, \tau_k^{min})$ to $(\tau_1^{min}, \tau_2^{min}, \ldots, \tau_k^{max})$
        check (14e) is feasible $\forall j = 1, 2, \ldots, k$ and whether (15) is satisfied
        if true **then**
            return $c_i\ \forall i = 0, 1, \ldots, r-1$; $\tau_j\ \forall j = 1, 2, \ldots, k$ and $\mu(x) = \lambda(x)$
        end if
    end loop
end loop

we have that $h(x) \geq 0$, $[u]_i - [u]_{l_j,i}^{min} \geq 0$ and $[u]_{l_j,i}^{max} - [u]_i \geq 0$. Due to (14a) and the condition that $q_j(x,u)$, $w_{j,i}(x,u)$, and $v_{j,i}(x,u)$ are SOS for all $i = 1, \ldots, m$ and $j = 1, 2, \ldots, k-1$, we have that for all $(x,u) \in C \times Inv_u(l_j)$ the following relation holds:

$$L_f^r h(x) + L_g L_f^{r-1} h(x) u - s_j \geq q_j(x,u) h(x) + \sum_{i=1}^{m} \Big(w_{j,i}(x,u)([u]_i - [u]_{l_j,i}^{min}) + v_{j,i}([u]_{l_j,i}^{max} - [u]_i) \Big) \geq 0,\ \forall j = 1, 2, \ldots, k-1.$$

Hence condition Equation (2a) holds. Now we will show that Equations (14b) and (14d) imply (2b). Consider $x \in C \backslash \mathcal{A}$. Then we have that $h(x) \geq 0$ and $(L_f^i h(x) - a_{0,i}) = (L_f^i h(x) - c_i) \leq 0\ \forall i = 0, 1, \ldots, r-1$. Using (14b) and the fact that $l(x,u)$ and $p_i(x,u)$ are SOS for all $i = 1, 2, \ldots, m$, for all $x \in C \backslash \mathcal{A}$ we can write

$$L_f^r h(x) + L_g L_f^{r-1} h(x) \lambda(x) - s_k \geq l(x) h(x) - \sum_{i=0}^{r-1} p_i(x)(L_f^i h(x) - a_{0,i}) \geq 0. \tag{16}$$

Thus condition (2b) holds as (14d) ensures that $[u]_{l_k,i}^{min} \leq \lambda_i(x) \leq [u]_{l_k,i}^{max},\ \forall i = 1, 2, \ldots, m$, i.e., $\mu(x) = \lambda(x) \in Inv_u(l_k)$. In the same manner, it can be shown that Equations (14c) and (14d) together imply (2c), and Equation (14e) implies (2d). Thus proof is completed. □

The above SOS formulation allows us to construct an algorithm for computing control policy and the other parameters as shown in Algorithm 1. In Algorithm 1, we search for $c_0, c_1, \ldots, c_{r-1}$ from $(0, 0, \ldots, 0)$ to $(c_0^{max}, c_1^{max}, \ldots, c_{r-1}^{max})$ where $c_0^{max}, c_1^{max}, \ldots, c_{r-1}^{max}$ respectively denotes selected upper bound on $h(x), h^1(x), \ldots, h^{r-1}(x)$ in the set C. The order and step sizes in the update of these parameters are chosen appropriately. Next, for the selected values of $c_0, c_1, \ldots, c_{r-1}$, we compute s_i for all $i = 1, 2, \ldots, s_k$ and $\mu(x)$ so that the conditions (14a), (14c) and (14d) are satisfied. Then we search for $(\tau_1, \tau_2, \ldots, \tau_k)$ from $(\tau_1^{max}, \tau_2^{max}, \ldots, \tau_k^{min})$ to $(\tau_1^{min}, \tau_2^{min}, \ldots, \tau_k^{max})$ and check whether the condition (15) is satisfied. If it is satisfied, obtained $\mu(x)$ is the desired control policy. But if (15) is not satisfied for any choice of $(\tau_1, \tau_2, \ldots, \tau_k)$, then $c_0, c_1, \ldots, c_{r-1}$ are updated until a solution is found or $(c_0^{max}, c_1^{max}, \ldots, c_{r-1}^{max})$ is reached. Since it is desired that system quickly returns to the level set $\mathcal{A}$, we initialize τ_k with τ_k^{min} unlike the other parameters $\tau_1, \ldots, \tau_{k-1}$.

Table 2. This Table Presents the Design Parameters to Compute in Different Resilient Architectures via Mapping to the Algorithm 1

CRAs	k	Sequence of Locations $l_1, l_2, \ldots, l_k$	Known Parameters	Design Parameters
BFT++ [27]	3	$corrupted, restoration, normal$	$\tau_1 = N_1\delta$, $\tau_2 = N_2\delta$	$\tau_3 = N_3\delta$, $\mu(x)$
YOLO [7, 8] / Dual Redundant [15]	2	$corrupted, restart/switching$		$\tau_1 = N_4\delta$, $\tau_2 = N_5\delta$
Proactive Restart [1, 39]	3	$corrupted, restart, SC$	$\tau_2 = N_7\delta$	$\tau_1 = N_6\delta$, $\tau_2 = N_8\delta$, $\mu(x)$
Reactive Restart [32]	3	$corrupted, restart, normal$		$\tau_1 = N_9\delta$, $\tau_2 = N_{10}\delta$, $\tau_3 = N_{11}\delta$, $\mu(x)$
Simplex/S3A [9, 29, 41]	1	SC		$\mu(x)$

For the resilient architectures in the first column, the corresponding numbers of discrete locations k and the corresponding sequences of discrete locations in each run are listed in the second and third column, respectively. The known parameters and the parameters to be designed in these architectures using the proposed algorithm are listed in the fourth and fifth column, respectively.

In Algorithm 1, we assume that all the timing parameters (e.g., $\tau_1, \tau_2, \ldots, \tau_k$) are unknown. If any of these parameters is known, we use that value in the algorithm without varying the parameter. The proposed algorithm is flexible in the sense that it can be mapped to a CPS employing any of the CRAs to design control policy and associated timing parameters. Table 2 provides a mapping to CRAs to solve various design problems inherent to the architectures. The last column in Table 2 lists the timing parameters to be computed for specific CRAs using our algorithm. We note that in addition to these parameters, we also need to compute the level set parameters $c_0, c_1, \ldots, c_{r-1}$. In this table, we consider the worst-case scenario in the sense that if it is not known whether the controller is compromised or not, then we assume the controller is compromised. For example, in the case of YOLO the system can be at either $l = normal$ or $l = corrupted$ location prior to restart. Therefore, we assume, only $l = corrupted$ and $l = restart$ for YOLO when designing the timing parameters to ensure that CPS remains safe in the worst-case scenario. This in fact is equivalent to letting $l_3 = normal$ and $\tau_3 = 0$ during design. The same analysis is applied to dual redundant and proactive restart schemes as it can be seen from the table. Note that for the class of Simplex architecture, there is no associated timing parameter. In this case, we only synthesize the control policy of the safety controller. For synthesizing the control policy, we only consider the conditions Equations (14c) and (14d). We remark that Algorithm 1 can also be used for safety verification of a given control policy $\psi(x)$ by checking the feasibility of the conditions in Proposition 5.5 for $\lambda(x) = \psi(x)$. If the conditions are feasible, then control policy $\psi(x)$ will guarantee safety.

Now, we will characterize the convergence of our algorithm. To do so, first we provide a justification of Line 5 of our algorithm where we maximize s_j subject to Equations (14a), (14b), (14c), and (14d) for $j = 1, 2, \ldots, k$. We show that maximizing s_j will suffice to find a solution to Proposition 5.5 if any solution exists. The result is formalized as below.

Lemma 5.6. *Suppose s_i^{max} maximizes s_j subject to Equation (14a) for $j = 1, 2, \ldots k-1$ and s_k^{max} maximizes s_k subject to Equations (14b), (14c), and (14d) for the selected $c_0, c_1, \ldots c_{r-1}$. If (14e) and (15) are not satisfied by $s_1^{max}, s_2^{max}, \ldots, s_k^{max}$ and the selected $\tau_1, \tau_2, \ldots, \tau_k$, then there exits no solution to the conditions given by Proposition 5.5 for the selected $c_0, c_1, \ldots c_{r-1}$ and $\tau_1, \tau_2, \ldots, \tau_k$.*

Proof. We will prove this lemma using contradiction. Suppose there exist $\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_k$ such that conditions given by Proposition 5.5 are satisfied for the selected $c_0, \ldots, c_{r-1}$ and $\tau_1, \tau_2, \ldots, \tau_k$. We now show that this implies $s_1^{max}, s_2^{max}, \ldots, s_k^{max}$ also satisfy Equations (14e) and (15). Since s_j^{max} maximizes s_j subject to (14a) for all $j = 1, 2, \ldots k-1$ and s_k^{max} maximizes s_k subject to Equations (14b), (14c), and (14d) for the selected $c_0, c_1, \ldots c_{r-1}$, therefore $\tilde{s}_j \leq s_j^{max}$ for all $j = 1, 2, \ldots, k$. Since according to (3b) each $a_{j,i}$ is a non-decreasing function of each s_j, the expressions in the left hand sides of Equations (2d) and (2e) are also non-decreasing functions of each s_j where $j = 1, 2, \ldots, k$ and $i = 0, 1, \ldots, r-1$. Hence $s_1^{max}, s_2^{max}, \ldots, s_k^{max}$ satisfy Equations (2d) and (2e). Since for univariate polynomial non-negativity and SOS conditions are equivalent [35], therefore, $s_1^{max}, s_2^{max}, \ldots, s_k^{max}$ also satisfy Equations (14e) and (15). But this is contradictory to the assumption pf the statement of lemma. Hence proof is complete. □

Now we present our main result on the convergence of Algorithm 1. Our insight is that if there exists any solution that satisfies the conditions in Proposition 5.5 strictly in the sense that SOS are nonzero in Equation (14) and the inequality Equation (15) is strict, then the solution lies within the interior of a feasible solution set. In that case, by choosing sufficiently small step size for the update of $c_0, c_1, \ldots c_{r-1}$ and $\tau_1, \tau_2, \ldots, \tau_k$, convergence of the algorithm to a feasible solution can be guaranteed. The following proposition formalizes this.

Proposition 5.7. *Suppose there exits a solution* $(\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_{r-1}, \hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_k)$ *such that Equations* (14) *and* (15) *are satisfied strictly with* $0 \leq c_i \leq c_i^{max}\ \forall i = 0, 1, \ldots, r-1$ *and* $\tau_j^{min} \leq \tau_j \leq \tau_j^{max}\ \forall j = 1, 2, \ldots, k$. *Then Algorithm 1 finds a feasible solution with* $0 \leq c_i \leq c_i^{max}\ \forall i = 0, 1, \ldots, r-1$ *and* $\tau_j^{min} \leq \tau_j \leq \tau_j^{max}\ \forall j = 1, 2, \ldots, k$ *in finite number of iterations if step sizes for updating* $c_0, c_1, \ldots, c_{r-1}, \tau_1, \tau_2, \ldots$ *and* τ_k *are chosen appropriately small.*

Proof. Since the solution $(\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_{r-1}, \hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_k)$ satisfies Equations (14) and (15) strictly, therefore, there exists a solution interval $\mathcal{I} \in \mathbb{R}^{r+k}$ with non-zero measure for which Equations (14) and (15) are feasible where $0 \leq c_i \leq c_i^{max}\ \forall i = 0, 1, \ldots, r-1$ and $\tau_j^{min} \leq \tau_j \leq \tau_j^{max}\ \forall j = 1, 2, \ldots, k$. Denote the lengths of $\mathcal{I}$ in c_i and τ_j are, respectively, $\tilde{c}_i$ and $\tilde{\tau}_j$ where $i = 0, 1, \ldots, r-1$ and $j = 1, 2, \ldots, k$. Let the update for c_i in Line 4 of Algorithm 1 be $\epsilon_{c_i} \in (0, \tilde{c}_i)$ and the update of τ_j in Line 6 of Algorithm 1 be $\epsilon_{\tau_j} \in (0, \tilde{\tau}_j)$ for all $i = 0, 1, \ldots, r-1$ and for all $j = 1, 2, \ldots, k$. Then applying Lemma 5.6, we get that Algorithm 1 will terminate with a feasible solution. Otherwise the interval $\mathcal{I}$ contains some infeasible solutions to Equations (14) and (15) since $\epsilon_{c_i} < \tilde{c}_i$ and $\epsilon_{\tau_j} < \tilde{\tau}_j$ for all $i = 0, 1, \ldots, r-1$ and for all $j = 1, 2, \ldots, k$. But this is contradictory to the definition of the solution interval. Hence, Algorithm 1 will terminate with a feasible solution. □

We now discuss the time complexity of the algorithm in terms of the number of SOS programs that must be solved as a function of the parameters. We note that Algorithm 1 consists of three different kinds of SOS programs: (i) maximize s_j subject to Equation (14a) $\forall j = 1, \ldots, k-1$ within $\mathcal{S}(x, u)$, (ii) maximize s_k subject to Equations (14b), (14c), and (14d) within $\mathcal{S}(x)$, and (iii) check feasibility of Equation (14e) $\forall j = 1, 2, \ldots, k$ within $\mathcal{S}(t)$. The time complexity of SOS programs is generally problem dependent. The factors that impact the complexity include the degree of the polynomials and the number of state variables. However, when the degrees of polynomials are fixed, the complexity grows polynomially with respect to the number of variables [44]. We suppose that the worst-case time complexity for each of the three kinds of SOS programs is $T_{sos}(\mathcal{S}(x, u))$, $T_{sos}(\mathcal{S}(x))$ and $T_{sos}(\mathcal{S}(t))$, respectively. Since t is scalar, $T_{sos}(\mathcal{S}(t))$ is in general much smaller than $T_{sos}(\mathcal{S}(x, u))$ and $T_{sos}(\mathcal{S}(x))$. In the worst case, when none of the timing parameters $(\tau_1, \ldots \tau_k)$ is known, the number of times the algorithm solves each of the three kinds of SOS programs is given by (i) $(\Pi_{i=0}^{r-1} \lfloor \frac{c_i^{max}}{\epsilon_{c_i}} \rfloor)(k-1)$, (ii) $(\Pi_{i=0}^{r-1} \lfloor \frac{c_i^{max}}{\epsilon_{c_i}} \rfloor)$ and (iii) $(\Pi_{i=0}^{r-1} \lfloor \frac{c_i^{max}}{\epsilon_{c_i}} \rfloor)\ (\Pi_{j=1}^{k} \lfloor \frac{\tau_j^{max} - \tau_j^{min}}{\epsilon_{\tau_j}} \rfloor)k$. Therefore,

the worst-case time complexity of Algorithm 1 is $(\Pi_{i=0}^{r-1} \lfloor \frac{c_i^{max}}{\epsilon_{c_i}} \rfloor)$ $((k-1)T_{sos}(\mathcal{S}(x,u)) + T_{sos}(\mathcal{S}(x)) + k(\Pi_{j=1}^{k} \lfloor \frac{\tau_j^{max} - \tau_j^{min}}{\epsilon_{\tau_j}} \rfloor) T_{sos}(\mathcal{S}(t)))$. However, in practice, the time complexity is less as some of the timing parameters are known or the algorithm might find a solution and terminate early.

6 CASE STUDY

This section presents a case study as a verification of our proposed framework. We consider two vehicles including a leading vehicle and a trailing vehicle. The vehicles are modeled as point mass and are assumed to be moving along a straight road. The trailing vehicle is equipped with anACC.

We denote the velocities of the leading and trailing vehicles as v_l and v_f, respectively. The distance between the vehicles is denoted as D. Let $x = [v_l, v_f, D]^T$ be the state variable. Then, the vehicles jointly follow the dynamics given below [6]:

$$\dot{x} \doteq \begin{bmatrix} \dot{v}_l \\ \dot{v}_f \\ \dot{D} \end{bmatrix} = \begin{bmatrix} a_l \\ -\frac{F_r}{m} \\ v_l - v_f \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \\ 0 \end{bmatrix} u, \tag{17}$$

where $u \in [-1, 1]$ is the control input to the trailing vehicle, $m = 1$ is the mass of the trailing vehicle, $a_l = 0.3$ is the acceleration of the leading vehicle, and $F_r = f_0 + f_1 v_f + f_2 v_f^2$ models the resistance incurred by the trailing vehicle. In this case study, we choose $f_0 = 0$, $f_1 = 1$, and $f_2 = 0.5$ and note that the dynamics Equation (17) is nonlinear. We assume that $x \in \mathcal{X}$, where $\mathcal{X}$ is a compact set given as $\mathcal{X} = \{x : \|x\|^2 \leq d\}$ with $d = 10$. We let the initial system state be $x_0 = [0, 0, 3]^T$. We suppose that the control input is updated every 0.1 second, i.e., with frequency 10Hz. Here, we consider a state-feedback controller where the system states are fully observable.

The trailing vehicle is required to satisfy a safety constraint modeled as $x_t \in C$ for all $t \geq 0$, where $C = \{x : h(x) = D - 2 \geq 0\}$. That is, the distance between the leading and trailing vehicles is no less than 2, and the trailing vehicle should stay behind the leading vehicle. Note that the system is of relative degree two (i.e., $r = 2$) under the safety constraint since $L_g h(x) = 0$ and $L_f L_g h(x) = -\frac{1}{m}$. Therefore, control synthesis algorithm given by Reference [31] is not applicable here.

In the remainder of this section, we evaluate a collection of architectures including BFT++ [27], YOLO [7, 8], proactive restart [1, 39], reactive restart [32], dual redundant [15], and Simplex [9, 29, 41] architectures under our proposed framework. We compute the set $\mathcal{A}$ along with the control policy μ and corresponding timing parameters, as shown in Table 2. We also provide a comparison among these architectures based on our case study.

6.1 Evaluation of BFT++

In what follows, we evaluate BFT++ using our proposed framework on the system Equation (17). We consider that the trailing vehicle is subject to a cyber attack. The attack exploits the vulnerability of the trailing vehicle's controller, which will trigger a controller crash. We consider the worst-case time consumption for controller crash and restoration, which are $N_1 = 2$ and $N_2 = 2$ epochs, respectively [27]. We assume that the trailing vehicle is equipped with a memory of length 2, which stores the delayed control inputs, so as to provide input signal to the vehicle during controller restoration.

By Table 2, the hybrid system traverses *corrupted*, *restoration*, and *normal* statuses during one attack cycle. Under this setup, we have that during one attack cycle, the controller will take 4 epochs to return to the normal status following an exploitation by the attacker. Using Algorithm 1, we obtain that $c_0 = 0.8$ and $c_1 = 0.1$. In addition, the controller should remain in *normal* status for at least $\tau_3 = 4$ epochs. We simulate the distance between the leading and trailing vehicles in Figure 4(a) by considering the worst-case scenario where the input signal during *corrupted* and

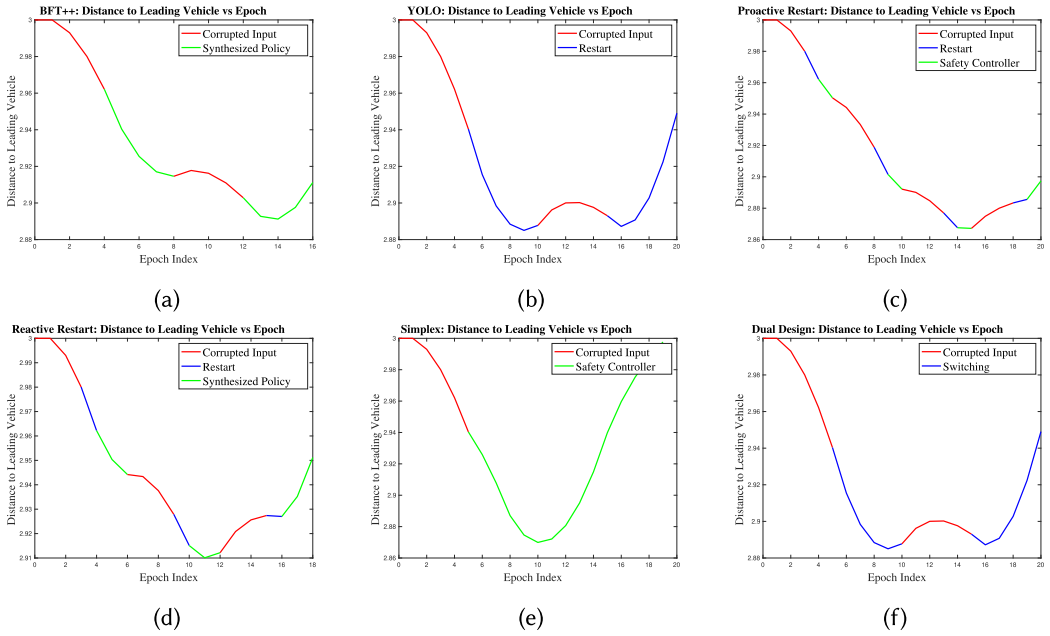

Fig. 4. This figure shows the distances between the leading and trailing vehicles in the case study where the trailing vehicle is subject to fault or attack. The trailing vehicle implements (a) BFT++, (b) YOLO, (c) proactive restart, (d) reactive restart, (e) Simplex, and (f) dual redundant scheme. The fragment of trajectory in red color indicates that the controller is corrupted. The blue color fragments capture the parts of trajectories generated when control input $u = 0$. The green color fragments depict the trajectories generated using the designed control policy or safety controller.

restoration statuses are both malicious. The trajectory generated by corrupted and delayed inputs is shown in red color, and the trajectory during the normal status is plotted in green color. Note that to guarantee safety of the system for possible future attack, the synthesized controller should ensure that $D \geq 2.8$ at the end of the normal status (as shown at epoch 8 and 16 in Figure 4(a)). In Figure 4(a), we observe that the adversary attempts to cause crash between the leading and trailing vehicles by injecting compromised inputs to decrease the distance between them (the red portion in Figure 4(a)). After the controller is restored, the designed control policy steers the trailing vehicle to set $\mathcal{A}$, as shown by the green portion in Figure 4(a). Note that because of Newton's law of motion there is a latency between distance D and the control input u in the dynamics (17). Due to the latency the distance continues to decrease under synthesized policy in the first green segment of Figure 4(a).

6.2 Evaluation of YOLO

This subsection evaluates YOLO [7, 8]. In this case, there is only one controller that turns on and off periodically. During one attack cycle, the hybrid system visits *corrupted*, *restart*, and *normal* statuses. We remark that, in this case, the trailing vehicle is not equipped with any mechanism to detect whether the system is compromised by the adversary or not. Therefore, in this case study, we will consider the worst-case scenario and assume that the controller produces compromised inputs when it is online. Thus, we will consider only *corrupted* and *restart* statuses for our design.

Using Algorithm 1, we have that $c_0 = 0.2$ and $c_1 = 0.1$. In addition, we obtain that $\tau_1 = \tau_2 = 5$ epochs. In other words, during one attack cycle, the controller is switched on and off every

5 epochs, yielding controller availability to be 50%. We plot the distance between the two vehicles in Figure 4(b). We use red and blue colors to represent the on (*corrupted*) and off (*restart*) statuses of the controller. We note that the adversary aims at causing vehicle crash by decreasing distance D, as shown by the red portion in Figure 4(b). However, the potential crash can be prevented by restarting the controller (the blue portion in Figure 4(b)). During the restart phase, the trailing vehicle slows down due to resistance, and thus maintains the distance to be within the safety set C.

6.3 Evaluation of Proactive Restart

In what follows, we evaluate the proactive restart based approach [1, 39]. We consider that the trailing vehicle is equipped with two controllers, one being main controller which is vulnerable to the cyber attack and the other one being the safety controller. After restart, the safety controller takes over the system to guarantee safety. To consider the worst-case scenerio, we assume that the main controller becomes compromised when it is online. However, safety controller does not become compromised since all the external interfaces are disabled during this interval [1, 39]. Hence, in this case, the system visits *corrupted*, *restart*, and *SC* statuses during one attack cycle.

Using Algorithm 1, we have that $c_0 = 0.83$ and $c_1 = 0.1$. Assuming that $\tau_2 = 1$ epoch, we also obtain the timing parameters as $\tau_1 = 3$ epochs and $\tau_3 = 1$ epoch. That is, the maximum number of epochs spent in *corrupted* and *restart* statuses are 3 and 1 epochs, respectively. The minimum number epochs in *SC* status is 1 epoch. We simulate the trajectory in Figure 4(c). We observe that the adversary can shorten the distance between the leading and trailing vehicles. The proactive restart commands sent by the decision module at epoch 3, 8, 13, and 18 prevent the adversary from further decreasing the distance between the vehicles. Furthermore, during the *restart*, since the leading vehicle's velocity is still less than that of the trailing vehicle, the distance between two vehicles continues to decrease, as shown by the blue fragment in Figure 4(c). During *SC*, the safety controller is implemented that uses the control policy synthesized according to our algorithm. This is shown by the green portion of the system trajectory in Figure 4(c) where we observe that the distance between the vehicles starts to increase so as to maintain the safety constraint and prevent safety violations in the future.

6.4 Evaluation of Reactive Restart

This subsection evaluates the reactive restart based approach [32]. In this setting, the trailing vehicle is equipped with one controller. Different from proactive restart based approach, restart command is issued as a response to adversarial exploitation, which triggers controller crash. In this case, the hybrid system visits *corrupted*, *restart*, and *normal* statuses during one attack cycle as shown in Table 2.

Using Algorithm 1, we obtain that $c_0 = 0.82$ and $c_1 = 0.1$. We also have that the controller tolerates $\tau_1 = 3$, $\tau_2 = 1$, and $\tau_3 = 2$ epochs in *corrupted*, *restart*, and *normal* statuses during one attack cycle. We simulate the distance between the vehicles in Figure 4(d). We observe that the adversary first reduces the distance between the two vehicles by accelerating the trailing vehicle, as shown by the red color fragment in Figure 4(d). Then the adversarial exploitation triggers controller restart and leads to input $u = 0$ (shown by the blue fragment in Figure 4(d)). After completing controller restart, the desired control input as computed using our algorithm is provided to the trailing vehicle, which increases the distance with the leading vehicle as seen in the green color fragment of Figure 4(d).

6.5 Evaluation of Simplex Architecture

This subsection evaluates Simplex architecture [9, 29, 41]. We consider that the trailing vehicle is equipped with a high performance controller and a safety controller. In this case study, we assume

that there exists no adversary. However, the high performance controller is vulnerable to random failures. The safety controller is assumed to be provably correct and thus is fault-free. Following the condition Equation (2c) and choosing $c_0 = 2.9, c_1 = 0.2$, we design the safety controller as

$$\min_{u} u^T u$$
$$\text{s.t. } L_f^2 h(x) + L_f L_g h(x) u + \alpha(L_f h(x) - 0.2) \geq 0$$

The trailing vehicle switches to the safety controller once the distance between two vehicles approaches to $D < 2.9$ or $h^1(x)$ approaches to $h^1(x) < 0.2$.

We simulate the trajectory in Figure 4(e). We observe that in the worst-case, the random failure reduces the distance between two vehicles and thus can potentially lead to vehicle crash. Once the condition for switching to safety controller is satisfied at epoch 5, the safety controller is invoked. From the green color fragment in Figure 4(e), we observe that the safety controller slows down the trailing vehicle, and thus increases the distance with the leading vehicle to prevent safety violation.

6.6 Evaluation of Dual Redundant Controllers Architecture

In this subsection, we consider that the trailing vehicle is equipped with two identical controllers [15] which are periodically switched to actuate the vehicle. However, the switching may not be instantaneous, and could incur some delay. During the delay, there is no control input to the system. As the worst-case scenario, we assume that the controllers are compromised when they are online. For this case, the hybrid system visits *corrupted* and *switching* statuses during one attack cycle.

Using Algorithm 1, we have that the system spends 5 epochs each for status *switching* and *corrupt*. We simulate the system trajectory as shown in Figure 4(f), which is similar to the one for YOLO i.e., Figure 4(b). We observe that if the system is compromised, then the distance between two vehicles decreases. However, switching between controllers ensures that the safety constraint is satisfied.

6.7 Comparison of Resilient Architectures

In this subsection, we compare the resilient architectures evaluated using our proposed framework considering the following aspects: (i) whether the system can recover from cyber attack, (ii) whether redundancy is required, (iii) the worst-case and best-case controller availabilities at each attack cycle, (iv) the design freedom, and (v) the maximum impact introduced by the adversary. The controller availability is measured T_{on}/T_{off}, where T_{on} and T_{off} are the number of epochs that some controller is online and offline during one attack cycle, respectively. We further divide the controller availability into two cases. The first case gives controller availability under attack which accounts the total amount of epochs when the controller is online, regardless of whether the controller being compromised or not. The second case gives the nominal controller availability, which only counts the epochs during which the controller follows the nominal policy into T_{on}. We consider the design freedom of an architecture to be the number of design parameters in our proposed framework as given in Table 2. We quantify the maximum impact introduced by the adversary as the maximum amount of decrease in the value of $h(x)$ (i.e., distance D between the leading and trailing vehicles) since the first epoch. The comparison result is summarized in Table 3. We observe that Simplex and S3A are not suitable for the system subject to cyber attacks. Among the CRAs, BFT++ provides the maximum availability of the nominal controller under attack during considered attack cycle. We also observe that reactive restart provides the least impact from the adversary (0.09) since it has four design parameters (τ_1, τ_2, τ_3 and $\mu(x)$) unlike other CRAs evaluated in this section.

Table 3. This Table Compares among the Existing Resilient Architectures when Applying to Adaptive Cruise Control Using Our Proposed Framework

CRAs	Cyber Attack	Redundancy	Controller Availability	Design Freedom	Maximum Impact
Simplex/S3A [9, 29, 41]	×	Redundant controller	100%(0%)	1	0.13
BFT++ [27]	✓	Redundant controllers	75%(50%)	2	0.1088
YOLO [7, 8]	✓	NA	50%(0%)	2	0.115
Dual Redundant [15]	✓	Redundant controller	50%(0%)	2	0.115
Proactive Restart [1, 39]	✓	Redundant control program	80%(20%)	3	0.1324
Reactive Restart [32]	✓	NA	83.33%(33.33%)	4	0.09

We compare the architectures in terms of (i) whether the system can recover from cyber attack (second column), (ii) whether redundancy is required (third column), (iii) the controller availability at each attack cycle (fourth column), (iv) the design freedom (fifth column), and (v) the maximum impact introduced by the adversary (sixth column). In fourth column, we list both the controller availability under attack and nominal controller availability, where the latter is presented in parenthesis. The architectures are listed in an ascending order of their respective the design freedom.

7 CONCLUSION

We developed a timing-based framework for safety analysis and parameter design of CPS that applies to a set of seemingly unrelated resilient architectures. We presented a hybrid system model to capture CPS employing distinct CRAs. We used the transition model of the hybrid system to derive architecture-agnostic sufficient conditions for control policy and timing parameters that ensures safety of the CPS. Our derived conditions hold for a system whose barrier certificate for safety is of higher relative degree with respect to the physical dynamics. We formulated the conditions as a SOS program, and based on that, we proposed an algorithm for computing control policy and timing parameters of the implemented architecture. Our derived conditions and proposed algorithm are flexible enough to map them into various design problems relevant to resilient architectures. We also analyzed the convergence of our algorithm and proved that the algorithm converges to a feasible solution under certain conditions. We presented a case study on the ACC of vehicles to demonstrate applicability of our proposed framework for different CRAs. As future directions of this work, we will develop algorithms to compute memory-dependent safe control policies. We are also interested in deriving an online setting for the proposed solution. We will also extend the proposed framework for interconnected CPS incorporating other system properties such as stability and reachability and the case where the state space can be mode dependent. Another possible direction of this work could be the development of a framework for synthesizing new and optimal CRAs.

REFERENCES

[1] Fardin Abdi, Chien-Ying Chen, Monowar Hasan, Songran Liu, Sibin Mohan, and Marco Caccamo. 2018. Guaranteed physical security with restart-based design for cyber-physical systems. In *ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS'18)*. IEEE, 10–21.

[2] Fardin Abdi, Rohan Tabish, Matthias Rungger, Majid Zamani, and Marco Caccamo. 2017. Application and system-level software fault tolerance through full system restarts. In *ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS'17)*. IEEE, 197–206.

[3] Amir Ali Ahmadi, Georgina Hall, Antonis Papachristodoulou, James Saunderson, and Yang Zheng. 2017. Improving efficiency and scalability of sum of squares optimization: Recent advances and limitations. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC'17)*. IEEE, 453–462.

[4] Homa Alemzadeh, Daniel Chen, Xiao Li, Thenkurussi Kesavadas, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. 2016. Targeted attacks on teleoperated surgical robots: Dynamic model-based detection and mitigation. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'16)*. IEEE, 395–406.

[5] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. 2019. Control barrier functions: Theory and applications. In *18th European Control Conference (ECC'19)*. IEEE, 3420–3431.

[6] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. 2016. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. Automat. Control* 62, 8 (2016), 3861–3876.

[7] Miguel Arroyo, Hidenori Kobayashi, Simha Sethumadhavan, and Junfeng Yang. 2017. FIRED: Frequent inertial resets with diversification for emerging commodity cyber-physical systems. arXiv:1702.06595. Retrieved from https://arxiv.org/abs/1702.06595.

[8] Miguel A. Arroyo, M. Tarek Ibn Ziad, Hidenori Kobayashi, Junfeng Yang, and Simha Sethumadhavan. 2019. YOLO: Frequently resetting cyber-physical systems for security. In *Autonomous systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2019*, Vol. 11009. International Society for Optics and Photonics, 110090P.

[9] Stanley Bak, Deepti K. Chivukula, Olugbemiga Adekunle, Mu Sun, Marco Caccamo, and Lui Sha. 2009. The system-level simplex architecture for improved real-time embedded system safety. In *15th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 99–107.

[10] Franco Blanchini and Stefano Miani. 2008. *Set-Theoretic Methods in Control*. Vol. 78. Springer.

[11] Alvaro A. Cárdenas, Saurabh Amin, and Shankar Sastry. 2008. Research challenges for the security of control systems. In *3rd Conference on Hot Topics in Security*, Vol. 5. USENIX Association, 15.

[12] Miguel Castro and Barbara Liskov. 2002. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)* 20, 4 (2002), 398–461.

[13] Max H. Cohen and Calin Belta. 2020. Approximate optimal control for safety-critical systems with control barrier functions. In *59th Conference on Decision and Control (CDC'20)*. IEEE, 2062–2067.

[14] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. 2014. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Transactions on Automatic Control* 59, 6 (2014), 1454–1467.

[15] Marco A. Gamarra, Sachin Shetty, Oscar R. Gonzalez, Laurent Njilla, Marcus Pendleton, and Charles Kamhoua. 2019. Dual redundant cyber-attack tolerant control systems strategy for cyber-physical systems. In *IEEE International Conference on Communications (ICC'19)*. IEEE, 1–7.

[16] Andy Greenberg. 2015. Hackers Remotely Kill a Jeep on the Highway–with me in it. Retrieved from https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.

[17] Paul Griffioen, Raffaele Romagnoli, Bruce H. Krogh, and Bruno Sinopoli. 2019. Secure networked control via software rejuvenation. In *2019 IEEE 58th Conference on Decision and Control (CDC'19)*. IEEE, 3878–3884.

[18] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. 2008. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *2008 IEEE Symposium on Security and Privacy (sp'08)*. IEEE, 129–142.

[19] Sylvia L. Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F. Fisac, and Claire J. Tomlin. 2017. FaSTrack: A modular framework for fast and guaranteed safe motion planning. In *56th Annual Conference on Decision and Control (CDC'17)*. IEEE, 1517–1522.

[20] Gaurav S. Kc, Angelos D. Keromytis, and Vassilis Prevelakis. 2003. Countering code-injection attacks with instruction-set randomization. In *10th ACM Conference on Computer and Communications Security*. 272–280.

[21] H. K. Khalil. 1996. Nonlinear systems, printice-hall. *Upper Saddle River, NJ* 3 (1996).

[22] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. 2010. Experimental security analysis of a modern automobile. In *2010 IEEE Symposium on Security and Privacy*. IEEE, 447–462.

[23] Per Larsen, Andrei Homescu, Stefan Brunthaler, and Michael Franz. 2014. SoK: Automated software diversity. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 276–291.

[24] M. Robert Lee, J. Michael Assante, and Tim Conway. 2016. Analysis of the Cyber Attack on the Ukrainian Power Grid. Retrieved from https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/05/20081514/E-ISAC_SANS_Ukraine_DUC_5.pdf.

[25] Mohammad Hossein Manshaei, Quanyan Zhu, Tansu Alpcan, Tamer Başar, and Jean-Pierre Hubaux. 2013. Game theory meets network security and privacy. *ACM Computing Surveys (CSUR)* 45, 3 (2013), 1–39.

[26] Tim Martin and Frank Allgöwer. 2021. Data-driven system analysis of nonlinear systems using polynomial approximation. arXiv:2108.11298. Retrieved from https://arxiv.org/abs/2108.11298.

[27] J. Sukarno Mertoguno, Ryan M. Craven, Matthew S. Mickelson, and David P. Koller. 2019. A physics-based strategy for cyber resilience of CPS. In *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2019*, Vol. 11009. International Society for Optics and Photonics, 110090E.

[28] Yilin Mo and Bruno Sinopoli. 2010. False data injection attacks in control systems. In *Preprints of the 1st Workshop on Secure Control Systems*, Vol. 1.
[29] Sibin Mohan, Stanley Bak, Emiliano Betti, Heechul Yun, Lui Sha, and Marco Caccamo. 2013. S3A: Secure system simplex architecture for enhanced security and robustness of cyber-physical systems. In *2nd ACM International Conference on High Confidence Networked Systems*. 65–74.
[30] Luyao Niu, Zhouchi Li, and Andrew Clark. 2019. LQG reference tracking with safety and reachability guarantees under false data injection attacks. In *2019 American Control Conference (ACC'19)*. IEEE, 2950–2957.
[31] Luyao Niu, Abdullah Al Maruf, Andrew Clark, J. Sukarno Mertoguno, and Radha Poovendran. 2022. An analytical framework for control synthesis of cyber-physical systems with safety guarantee. In *IEEE 61st Conference on Decision and Control (CDC'22)*.
[32] Luyao Niu, Dinuka Sahabandu, Andrew Clark, and Poovendran Radha. 2022. Verifying safety for resilient cyber-physical systems via reactive software restart. In *To Appear in ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS'22)*. ACM/IEEE.
[33] Miroslav Pajic, Zhihao Jiang, Insup Lee, Oleg Sokolsky, and Rahul Mangharam. 2014. Safety-critical medical device development using the UPP2SF model translation tool. *ACM Transactions on Embedded Computing Systems (TECS)* 13, 4s (2014), 1–26.
[34] Miroslav Pajic, James Weimer, Nicola Bezzo, Paulo Tabuada, Oleg Sokolsky, Insup Lee, and George J. Pappas. 2014. Robustness of attack-resilient state estimators. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'14)*. IEEE, 163–174.
[35] Victoria Powers. 2011. Positive polynomials and sums of squares: Theory and practice. *Real Algebraic Geometry* 1 (2011), 78–149.
[36] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. 2007. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. Automat. Control* 52, 8 (2007), 1415–1428.
[37] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. 2021. Learning safe multi-agent control with decentralized neural barrier certificates. arXiv:2101.05436. Retrieved from https://arxiv.org/abs/2101.05436.
[38] Raffaele Romagnoli, Paul Griffioen, Bruce H. Krogh, and Bruno Sinopoli. 2020. Software rejuvenation under persistent attacks in constrained environments. *IFAC-PapersOnLine* 53, 2 (2020), 4088–4094.
[39] Raffaele Romagnoli, Bruce H. Krogh, and Bruno Sinopoli. 2019. Design of software rejuvenation for cps security using invariant sets. In *2019 American Control Conference (ACC'19)*. IEEE, 3740–3745.
[40] Raffaele Romagnoli, Bruce H. Krogh, and Bruno Sinopoli. 2020. Robust software rejuvenation for CPS with state estimation and disturbances. In *2020 American Control Conference (ACC'20)*. IEEE, 1241–1246.
[41] Lui Sha. 2001. Using simplicity to control complexity. *IEEE Software* 18, 4 (2001), 20–28.
[42] Farid Sharifi, Mostafa Mirzaei, Brandon W. Gordon, and Youmin Zhang. 2010. Fault tolerant control of a quadrotor UAV using sliding mode control. In *Conference on Control and Fault-Tolerant Systems (SysTol'10)*. IEEE, 239–244.
[43] Chuanzheng Wang, Yiming Meng, Yinan Li, Stephen L. Smith, and Jun Liu. 2021. Learning control barrier functions with high relative degree for safety-critical control. In *2021 European Control Conference (ECC'21)*. IEEE, 1459–1464.
[44] Tichakorn Wongpiromsarn, Ufuk Topcu, and Andrew Lamperski. 2015. Automata theory meets barrier certificates: Temporal logic verification of nonlinear systems. *IEEE Trans. Automat. Control* 61, 11 (2015), 3344–3355.
[45] Dezhi Xu, Fanglai Zhu, Zepeng Zhou, and Xinggang Yan. 2020. Distributed fault detection and estimation in cyber–physical systems subject to actuator faults. *ISA Transactions* 104 (2020), 162–174.
[46] Xiangru Xu. 2018. Constrained control of input–output linearizable systems using control sharing barrier functions. *Automatica* 87 (2018), 195–201.
[47] Youmin Zhang and Jin Jiang. 2008. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control* 32, 2 (2008), 229–252.
[48] Quanyan Zhu and Tamer Başar. 2015. Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: Games-in-games principle for optimal cross-layer resilient control systems. *IEEE Control Systems Magazine* 35, 1 (2015), 46–65.

Received 28 August 2022; revised 8 March 2023; accepted 11 April 2023