# Securing Cyber-Physical Systems with Adaptive Commensurate Response

Zhiyuan Zheng*, Shan Jin*, Riccardo Bettati† and A. L. Narasimha Reddy*
* Department of Electrical & Computer Engineering, Texas A & M University
†Department of Computer Science & Engineering, Texas A & M University
Email: { zhiyuanbj, jinshan, bettati, reddy }@tamu.edu

***Abstract*—Cyber-Physical Systems (CPSs) are ubiquitous in power systems, transport and medical systems, and critical infrastructures. Current protocols and applications allow significant changes to a system to take place within a short time or small network footprint, which can be exploited by attackers to cause a great impact on the physical systems. This paper proposes adaptive Commensurate Response (CR) to narrow down the asymmetry between the cost of attacks and their impact through enforcing command footprints to be commensurate with their impact on the system. Such impact is measured by the change of the setpoint (change-driven CR) or the distance between the operating state and the critical state (criticality-driven CR). Change-driven CR is effective for setpoint attacks with big setpoint changes, and criticality-driven CR can be used to combat both setpoint attacks and actuation attacks. Our case study on automobile cruise control demonstrates that CR can effectively improve the system resilience and attack survivability while satisfying QoS requirements.**

***Keywords*—*Cyber-Physical Systems, SCADA, Commensurate Response, Industrial Control Systems, PLC***

## I. Introduction

CYBER-PHYSICAL Systems (CPSs) have become the core components of safety-critical infrastructures such as smart grid [1], building automation networks [2] and water/sewage plants [3]. CPSs bridge the cyber and the physical world by integrating the computing and communication capabilities of the former to monitor, model, control, and analyze processes in the latter. Due to the criticalness of the infrastructures in which CPSs are deployed, they have become a ripe target for cyber-attacks. Examples include Stuxnet [4], which attacked Iran's nuclear centrifuges by sending malicious commands to periodically modify their motor's spinning frequency while reporting normal sensing values. Similarly, hackers caused the massive Ukraine power outage [5] on Dec. 23rd, 2015, by sabotaging the control system and remotely opening the breakers. This incident affected about 230,000 people and was regarded as the first power outage caused by hackers. Often, the security protocols deployed in such systems are not corresponding with their criticality.

Many cyber-physical protocols and applications allow for situations where a single event can have a significant impact on the system, *e.g.* a single message from a controller may reset the temperature control of entire buildings. Similarly, a small burst of malicious actuation messages over a CAN bus in an automotive cruise control setting can cause the vehicle to operate at dangerous speed. In these examples, the adversaries leverage the inherent asymmetry between small effort and costly consequence to launch attacks that are either difficult to detect or difficult to counter in real time: the adversary can mount a small-footprint attack (*i.e.*, with a very small number of commands so that it cannot be discerned from nominal network traffic) that gives rise to difficult-to-detect anomalous behavior that in turn can cause a significant impact [4]. This renders the design of effective and non-intrusive countermeasures particularly difficult. In real-time scenarios, where timely delivery of messages is critical, the requirement to not unduly delay the delivery of messages (due to false positives or due to the need to collect authorization information) further complicates the design of protective countermeasures.

Research work in protecting CPS can be classified into two categories: communication networks approaches and system-theoretical approaches. The former category focus on the SCADA network and try to prevent malicious activities through methods such as IDS/anomaly detection [6]–[10], authentication [11]–[14], and encryption [15]. System-theoretical approaches, on the other hand, focus on the control system itself. Such protections are usually based on the inherent control plane dependencies and physical relationships. For example, dynamic watermarking [16]–[18] is effective to detect a number of stealty attacks including False Data Injection Attacks [19]–[22] and replay attacks [23].

In this paper we present a framework, which we call Commensurate Response (CR), to harden CPS systems by addressing and reducing the asymmetry between the low cost of launching an attack using spurious and difficult-to-detect control messages and the costly effect that such an attack can have on the system. CR forces the footprint of the attack to be commensurate with its impact on the system. In some cases, this can be achieved by limiting the changes in actuation that can be triggered by a command. In order to initiate drastic changes in the actuation, CR would require the attacker to issue multiple, consistent requests. The larger the intended change, the larger the number of requests to be sent. This in turn increases the time for defensive measures, such as intrusion detection systems (IDSs), to detect the attack and to respond.

This paper describes how to add CR capabilities to existing systems. We discuss where CR modules are to be added, and how they are to be designed and parameterized in order to protect the system without violating its original design requirements. We describe two types of CR designs, which we call *change-driven* CR and *criticality-*

*driven* CR. Both designs provide commensurate response by dynamically controlling the system response latency as a function of current risk exposure and defense posture. In the case of change-driven CR, the risk exposure is measured in terms of the degree at which an issued command changes the plant operating setpoint. Given that an attacker may be masquerading as operator, large changes to the operating setpoint must be treated with suspicion, and thus the system response latency has to be temporarily increased to allow for defensive measures to react. For commands with a drastic change in setpoint, change-driven CR requires a longer response time for the plant to reach the setpoint, or the sender needs to issue multiple requests with smaller setpoint change. Instead of a single message to increase the room temperature by 20 Celsius degree within one minute, attackers would now need to send 20 consistent messages or wait several minutes before reaching the setpoint. The additional footprints would greatly help the IDS to identify suspicious activities. Criticality-driven CR, on the other hand, monitors the system state and measures risk exposure as the current distance of the current operating point to the critical point, As the plant approaches a critical point, the CR module gradually tunes the system response to prevent the plant from reaching the critical point. In both cases, CR improves the system resilience and survivability [24] while guaranteeing the QoS (quality-of-service) and without affecting normal operations. We will describe how CR complements other defense mechanisms such as dynamic watermarking and deep packet inspection (DPI) to better secure CPS.

To evaluate our security mechanism, we provide a case study on automotive cruise control over a CAN bus and evaluate the performance of the two CR schemes. Results show that change-driven CR is effective when the attacker aims at manipulating the system setpoint (so-called setpoint attacks), and criticality-driven CR can be used against setpoint attacks and against situations when the attacker manipulates the system actuators (so-called actuation attacks). The benefits of our solution are the following. First, when the attacker tries to drive the system into a dangerous state, CR can effectively change the current state from unsafe to safe. Next, the CR module modifies system dynamics and expands acceptable operation range to accommodate different application needs. Finally, CR improves the system resilience and survivability and prevents stealthy attacks that cannot be detected by network-level protections (*e.g.* deep packet inspection).

We summarize the contribution of this paper as follows:

- We propose Commensurate Response (CR) as a security mechanism that operates within the control domain to improve the security of a CPS while maintaining its operational constraints. CR complements other security mechanisms, such as network-level IDSs.
- Change-driven CR brings IDS latency into system design consideration and enforces more effort for commands with larger setpoint change. It makes it difficult for attackers to cause drastic changes to a CPS.
- Criticality-driven CR tunes the system response as a function of how far the plant is from the critical point. This form of CR can be used against setpoint attacks and actuation attacks.
- We evaluate our solution using an automobile cruise control system operating over a CAN bus. We expose the system to attacks and demonstrate that CR improves the system resilience and attack survivability.

The rest of the paper is organized as follows. Section II formulates the problem and introduces main components in a CPS. Section III discusses the attacker model. Section IV provides a detailed system-theoretical study of both change-driven CR and criticality-driven CR. An evaluation with a case study is presented in Section V. We make comparisons with related work in Section VII. Finally, we present our conclusions and lay out a path for future work in Section VIII.

## II. Problem Formulation

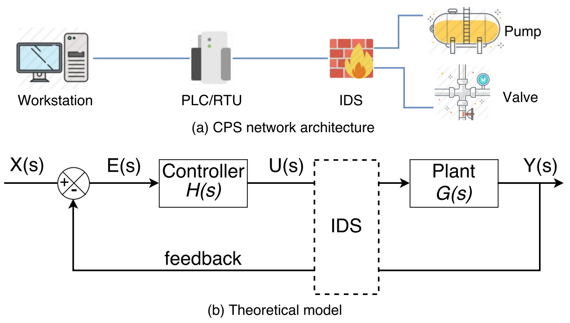

**Fig. 1:** CPS network architecture and the theoretical model.

Fig. 1 depicts a typical CPS network architecture in an industrial control setting and its corresponding theoretical control model. As shown in Fig. 1(a), such a system may consist of a workstation, one or more PLCs (programmable logic controllers) or RTUs (remote terminal units), an IDS (intrusion detection system), and physical devices such as water pumps, valves, and temperature sensors. The PLCs and RTUs take control commands from the workstation and perform local sensing and actuation on the connected physical devices, in this case a pump and a valve. The following discussion applies to CPSs beyond traditional industrial control systems. For example, in Section V we will discuss the application of CR in an automotive cruise control setting.

The canonical model for such a control system contains three main components: a *physical plant*, a *controller*, and a *feedback loop*, see Fig. 1(b). The physical plant $G(s)$ refers to the system process. It takes one or more actuation signals $U(s)$ from the controller, and generates sensing values $Y(s)$ as output signals. The output signals are subsequently compared with the setpoint values $X(s)$ through the feedback loop to compute the error signal $E(s) = X(s) - Y(s)$, that is, the deviation of the output from the setpoint. The controller interprets $E(s)$ and adjusts the discrepancy by either attenuating or amplifying the input signal to the plant.

The communication interfaces between the controller and the plant can be of different forms: wired and wireless, Ethernet and serial. In order to protect the plant, IDSs and alert systems may be installed to monitor both the actuation signal $U(s)$ and the sensing signal $Y(s)$ on the communication paths between the controller and the plant. Alerts are issued whenever the plant approaches a *critical* *point*, that is, a state at which the operation of the system becomes either detrimental to the plant operation or even dangerous. Operating the plant at or above the critical point must be avoided.

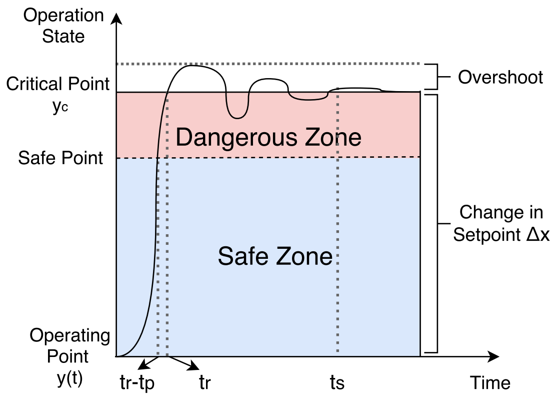

**Fig. 2:** A sample plant step response $y(t)$.

In order to simplify the following discussion, we describe our approach using a simple system, consisting of a linear time-invariant (LTI) plant that is controlled by a PID controller. This simple system allows us to identify and focus on a small number of easily quantifiable objectives: system stability, rise time, overshoot, and settling time. This choice of system is not limiting: PID controllers are commonly used in a variety of settings, including temperature control, automotive cruise control, and the Boeing 747 autopilot control system [25]. CR can also be easily extended to more general systems, but parameters and objectives would become more cumbersome to describe.

Fig. 2 shows the plant step response diagram where the X-axis denotes time and the Y-axis denotes the state $y(t)$ of the system. Four major characteristics are used to describe the system response performance of PID controlled systems: (1) *rise time* $t_r$ normally represents the amount of time the system takes to go from (0-10)% to (90-100)% of the targeted steady-state (we use the range of 1-99% in our study); (2) *settling time* $t_s$ represents the time it takes for the system to converge to the targeted steady state; (3) *overshoot* denotes how much the peak level is higher than the targeted steady state, normalized against the value of the latter; and (4) *steady-state error* denotes the difference between the actual steady state and the targeted steady state.

In addition, we define the *IDS response latency* $t_p$ as the time for the IDS to detect an attack and to respond to it. The value for $t_p$ is fixed for a given IDS. We note that for PID controllers in general the value for $t_r$ is not dependent on $\Delta x$ or the current operating point $y(t)$. Therefore we can say that ($t_r$ - $t_p$) defines the largest safe operating point, that is, the operating point closest to the critical point at which the IDS can still detect and recover from an attack that intends to move the plant over the critical point. We call this point the *safe point*. The safe point divides the operation range, as shown in Fig. 2, into the *safe zone*, from where the IDS can recover from an attack, and the *dangerous zone*, where the IDS does not have sufficient time to recover before the plant reaches the critical point.

Consider a hypothetical case where a cruise control system with a PID controller increases the vehicle speed from $y(t) = 10$ mph to 100 mph (the critical point). If the rise time is $t_r = 1.5s$ and the IDS latency $t_p = 1s$, alerts will be issued at $(t_r - t_p) = 0.5s$, and the safe point is 85 mph. If the rise time is less than $t_p$, say $t_r = 0.5s$, then the IDS cannot recover from in time: When the IDS detects the attack, the vehicle speed has already exceeded 100 mph. Thus alerts should be issued at or before time ($t_r$ - $t_p$). This requires the rise time $t_r$ to be larger than or equal to $t_p$.

In many CPS networks, the requirement for IDSs to recover in a timely fashion is not considered when deciding the control parameters. In addition, given the plant and PID parameters, the rise time $t_r$ is fixed and not dependent on the current operating point $y(t)$ or the setpoint change. Therefore, malicious control commands from the attacker can easily drive the system into a critical state within a short time, jeopardizing the safety and security of the system. The objective of CR is to actively manage the system response to maximize the safe operating range of the plant while at the same time maximizing the effort needed for an attacker to force the plant to or beyond its critical operating point.

Following textbook control theory, we represent a PID controller in the continuous-time domain as follows:

$$u(t) = K_p \cdot e(t) + K_i \int e(t)\mathrm{d}t + K_d \cdot \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{1}$$

where $u(t)$ denotes the control signal to the plant, $e(t) = x(t) - y(t)$ the control error, $x(t)$ the setpoint, and $y(t)$ the measured output from the plant. The control parameters are proportional gain $K_p$, integral gain $K_i$ and derivative gain $K_d$. In the Laplace domain, the transfer function of the PID controller $H(s)$ becomes:

$$H(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d \cdot s \tag{2}$$

where $s$ is the complex frequency.

We derive the close-loop transfer function $\frac{Y(s)}{X(s)}$ of the overall system as:

$$Y(s) = G(s) \cdot U(s) \tag{3}$$
$$U(s) = H(s) \cdot E(s) = H(s) \cdot (X(s) - Y(s)) \tag{4}$$
$$\frac{Y(s)}{X(s)} = \frac{G(s) \cdot H(s)}{1 + G(s) \cdot H(s)} \tag{5}$$

In practice, the system designer chooses the PID parameters ($K_p$, $K_i$, and $K_d$) using one of several methods, such as Ziegler–Nichols [26] or Tyreus-Luyben [27]. Usually, the resulting PID parameters are improved through additional tuning to meet the following design constraints:

1) The rise time $t_r$ is upper bounded by the system responsiveness requirement ($t_r \leq t_{max}$), especially for systems that consider responsiveness as a functional requirement (such as systems with hard deadlines);
2) Constraints on system *stability* (poles on the Left-Hand-Plane), *controllability* (rank issue), and *observability* (rank issue) need to be satisfied.

3) Application-specific requirements such as rise time, settling time, overshoot, and steady-state error need to be satisfied as well. Specific constraint values for these parameters vary across different applications and systems.

| Response | Rise Time | Overshoot | Settling Time | S-S Error |
|---|---|---|---|---|
| $K_p$ | Decrease | Increase | NT | Decrease |
| $K_i$ | Decrease | Increase | Increase | Eliminate |
| $K_d$ | NT | Decrease | Decrease | NT |

**TABLE I:** The effects of increasing each of $Kp, Ki, Kd$. NT means not definite trend. [28]

Table I illustrates how PID parameters related to system dynamics. For example, in order to increase the rise time $t_r$, we can decrease $K_p$ or $K_i$. However, the change in PID parameters will also result in the change in overshoot, settling time, and steady-state error. Similar tables are commonly used as guidelines for further PID tuning.

## III. Attacker Model

Attackers who have gained access to a CPS may launch attacks from different locations in the control loop. We identify two types of attacks: setpoint attacks and actuation attacks. Fig. 3 demonstrates the attacker model considered in our paper. We assume that attacks are perpetrated by insiders who either have direct access to the CPS network or who are capable of compromising devices in the CPS. They may attack the operator's workstation where the setpoint is issued, the controller, sensors, or communication channels. We do not consider attacks on sensors in this paper and assume that they function correctly. Solutions for tolerating attacks on sensors have been previously proposed in [12], [13].

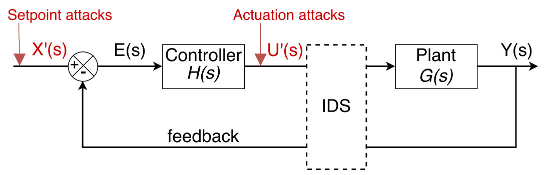

**Fig. 3:** The attacker model considered in our paper.

**Setpoint attacks.** Attackers may inject malicious commands that trigger big changes to the setpoint values issued to the controller. The objective could be to change the setpoint to drive the system into a critical state. Attackers may achieve this through compromising the operator's workstation or the HMI (human-machine interface), or by hijacking the existing communication session between the workstation and the controller. If unchecked, the corrupted setpoint signal $X'(s)$ is then executed by the controller.

**Actuation attacks.** Rather than changing the setpoint, an attacker may directly send malicious actuation commands to the plant on behalf of the controller, and so drive the system into a critical state. This attack can be achieved by (1) compromising the existing controller; by (2) hijacking the controller-plant communication channel and launching a man-in-the-middle attack; or by (3) having unauthorized devices masquerade as *bona fide* controllers and issue malicious actuation commands. We assume that the IDS monitoring points are judiciously placed, and malicious actuation commands cannot be inserted between the IDS and the actuators in the plant. Thus, the IDS monitors and processes any corrupted actuation signal $U'(s)$.

## IV. Adaptive Commensurate Response

In order for the IDS to respond in time, $t_r \geq t_p$ must hold. If this is not the case, the system designer has two options: replace the IDS by one with smaller detection latency and so decrease $t_p$; or change the plant dynamics to control the response time $t_r$. Often, intrusion detection in CPSs depends on correlation of actuation and sensing data over time, and so sufficiently reducing $t_p$ may not always be feasible. We therefore choose to explore modification of $t_r$ in the control plane. We are interested in controlling the system response under certain scenarios, so that the IDS and system operators have more time to detect suspicious packets when needed and as a result can take remedies accordingly.

Controlling the system response, and as a result modifying $t_r$, naturally controls how quickly a change of the plant state can take place as a result of the control input or of actuation commands. Attackers who attempt to circumvent CR have to send either more commands or commands with more aggressive setpoint or actuation requests. These actions will generate unusual spikes in the network traffic or control traffic with anomalous command sequences, both of which are easy to detect by the IDS.

### A. Methodology

Fig. 4 shows the system model with Commensurate Response where the CR module $C(s)$ is inserted in the forward path. In practical cases, $C(s)$ can also be inserted in the reverse path, and the theoretical analysis remains the same. The CR module $C(s)$ can be designed to operate in two modes: *conservative mode* and *aggressive mode*.

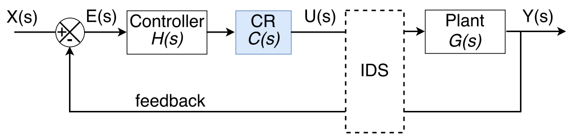

**Fig. 4:** System model with adaptive Commensurate Response.

**Conservative-Mode CR.** CR in conservative mode modifies the original signal on the time axis by either adding a *time delay* or a *time scaling* factor the original signal. The time delay module postpones the signal by a constant time $\tau$; and the time scaling module scales the signal on the time axis by a factor $a$. In discrete time, where signals are represented in the form of individual packets, the time scaling module can be implemented in two different ways: (1) fix packet count while increasing interpacket arrivals; or (2) fix interpacket arrivals while requiring more packets to be sent for a given action. We assume that the additional packets and latency have little impact on the bandwidth or network performance, and normal operations will not be affected.

**Aggressive-Mode CR.** When operating in aggressive mode, the CR module is designed so as to essentially replace the original controller with a new one that has the same structure but different control parameters. By selecting the parameters of the CR module the designer can control the system response while maintaining any desired system properties. We focus on this mode in the following discussion.

A PID controller can be represented as:

$$H(s) = \frac{K_p \cdot s + K_i + K_d \cdot s^2}{s} \tag{6}$$

We design $C(s)$ as:

$$C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{K_p \cdot s + K_i + K_d \cdot s^2} \tag{7}$$

The new PID controller $H(s) \cdot C(s)$ turns out to be:

$$H(s) \cdot C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{s} \tag{8}$$

$K_p^{cr}$, $K_i^{cr}$, and $K_d^{cr}$ become the new PID controller parameters. The parameters should be chosen properly to meet the design requirements discussed in Section II. $K_p^{cr}$, $K_i^{cr}$, and $K_d^{cr}$ now become functions of the *scheduling variable* $v(t)$:

$$K_p^{cr} = f_p(v(t)), \quad K_i^{cr} = f_i(v(t)), \quad K_d^{cr} = f_d(v(t)) \ .$$

We consider two different scheduling variables: (1) $\Delta x$: the change of the setpoint between two consecutive setpoint packets; and (2) $|y(t) - y_c|$: the distance between the current sensing value $y(t)$ and the system critical state $y_c$. CR that uses $\Delta x$ is called "change-driven CR" because it relies on the degree of setpoint change. CR that uses $|y(t) - y_c|$ is called "criticality-driven CR" because it is based on the closeness of the current operating point to the critical point. Both CR strategies turn the control problem from linear to non-linear. CR module changes $t_r$ from a fixed value to a function of $v(t)$. Instead of directly solving the non-linear problems, we employ *gain scheduling* to convert them into a set of linear problems, as discussed below.

## B. Change-driven Commensurate Response

The idea of change-driven CR is to (1) make sure that $t_r \geq t_p$ and (2) increase $t_r$ (slow down the system response) when a drastic change to the operating setpoint is about to take place. This mechanism not only provides enough time for the IDS to detect suspicious packets, but also enforces more effort when a significant change is about to take place in the system. Thus, attackers who aim to cause a big change to a system are forced to slow down by either waiting for a longer $t_r$ or sending more setpoint commands, with smaller setpoint changes. The additional response latency or messages will greatly facilitate the IDS to identify suspicious activities. In this case, we consider attackers may launch setpoint attacks, but not actuation attacks.

The transition of $t_r$ from a static value to a function of $\Delta x$ makes the size of dangerous zone to be dependent on $\Delta x$. As is shown in Fig. 5(a) and 5(b), change-driven CR is able to slow down the system response and extend the range of the safe zone while keeping the same time ($t_p$ in this case) to reach the setpoint (the critical point here). Change-driven CR can change the current operating point from dangerous to safe and prevent drastic changes in a system. When a new setpoint arrives, change-driven CR will be triggered and the control parameters will be adjusted. The control parameters remain consistent as the plant approaches the setpoint.

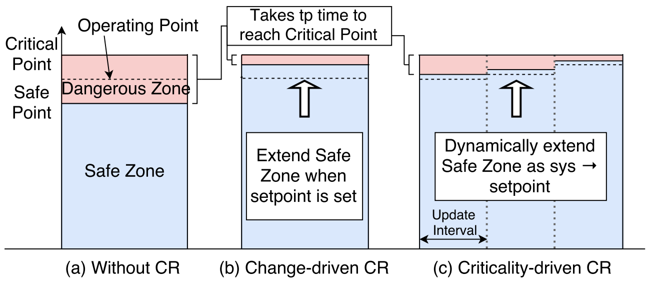
Operating Point
Critical Point
Safe Point
Dangerous Zone
Safe Zone
Takes tp time to reach Critical Point
Extend Safe Zone when setpoint is set
Dynamically extend Safe Zone as sys → setpoint
Update Interval
(a) Without CR
(b) Change-driven CR
(c) Criticality-driven CR

**Fig. 5:** System effect of adaptive Commensurate Response.

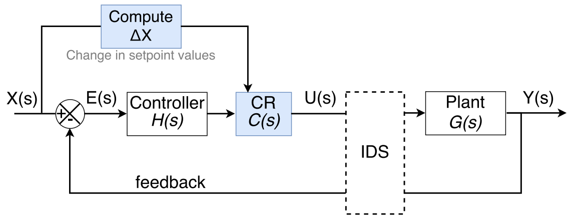

**Fig. 6:** System model for change-driven CR.

Fig. 6 shows the theoretical model for change-driven CR. When a new setpoint command arrives, the absolute value of the setpoint change between two consecutive messages ($\Delta x$) is computed and used by the CR module to decide the new $t_r$. Given the dynamics of a plant, design steps for change-driven CR are as follows:

*1)* Starting from the original controller whose parameters ($K_p$, $K_i$, and $K_d$) are determined by the existing PID tuning algorithms, measure the original rise time $t_r^0$.

*2)* Divide the largest possible setpoint range [0, $\Delta x^{max}$] into several intervals; design scheduling policy to match $\Delta x^i$ ($i^{th}$ interval of [0, $\Delta x^{max}$]) to a rise time $t_r^i$ ($t_r^i \in [t_p, t_{max}]$). A larger $\Delta x^i$ corresponds to a larger $t_r^i$.

*3)* For each $t_r^i$, obtain a set of PID parameter tuples that satisfy the design requirements.

*4)* Use greedy algorithm to select a PID parameter tuple for $t_r^i$. We select the one that has the smallest geometrical distance to the selected PID parameter tuple of $t_r^{(i-1)}$.

*5)* A scheduling table is created which projects $\Delta x^i$ to a rise time $t_r^i$ and further to a PID parameter tuple.

*6)* When a $\Delta x$ arrives, the CR module looks up the scheduling table, locates the $\Delta x$ interval, and chooses the PID parameters as $K_p^{cr}$, $K_i^{cr}$, and $K_d^{cr}$. $C(s)$ can be designed as $C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{K_p \cdot s + K_i + K_d \cdot s^2}$ and inserted into the CPS.

Even though the CR module resides on the actuation path in the theoretical model, in practical cases, $C(s)$ can be applied to the actuation signal $U(s)$, or the sensing signal $Y(s)$, or both.

## C. Criticality-driven Commensurate Response

In criticality-driven CR, the distance between the current operating point and the critical point, $|y(t) - y_c|$, is selected as the scheduling variable $v(t)$. Criticality-driven CR enforces state-based protection so that a *smaller* distance to the critical point results in a *larger* response time $t_r$. Attackers who aim to sabotage the plant will find it difficult to drive the system into a critical state. Both setpoint and actuation attacks result in the change of $y(t)$, so criticality-driven CR (triggered by $|y(t) - y_c|$) is applicable to both attacks. Criticality-driven CR improves the system resilience and attack survivability from the system-theoretical perspective.

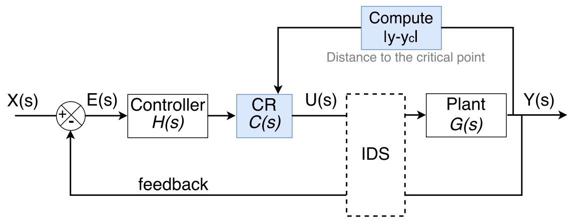

**Fig. 7:** System model for criticality-driven CR.

Fig. 7 shows the system model for criticality-driven CR. Different from change-driven CR, criticality-driven CR increases the resistance of the system response as the plant gets closer to the critical point $y_c$. As is shown in Fig. 5(c), criticality-driven CR gradually slows down the system response (expands the safe zone) as the system approaches the critical point. Since criticality-driven CR relies on $y(t)$ to decide $t_r$, we assume that $y(t)$ is valid and cannot be modified by the attackers.

The size of the dangerous zone in Fig. 5(c) is continuously changing, but the time to reach the critical point, namely $t_p$ in this case, remains constant. Criticality-driven CR essentially corresponds to the "two-second rule" in defensive driving. The two-second rule suggests a driver should ideally stay at least two seconds behind any vehicle that is directly in front of his or her vehicle, keeping a safe distance at any speed. The controlling variable is the vehicle speed and the dangerous zone is the safe distance to the car in front. Criticality-driven CR gradually slows down the vehicle speed as it gets closer to the car in front, while maintaining the two-second rule. Our model provides the theory behind the two-second rule and can be applied to many other application scenarios. The design steps for criticality-driven CR are similar to the ones for change-driven CR except that state information is fed into the CR module and a *smaller* $|y(t) - y_c|$ corresponds to a *larger* $t_r$.

Compared to the existing practices where a fixed "safe-range" is enforced at the firewalls or the IDS, our solution aids in the choice of "safe-range" value and makes the choice of safe point more dynamic and flexible to accommodate different application and operating scenarios. Both change-driven CR and criticality-driven CR can either be applied separately or at the same time in a CPS, which is further discussed in Section V.

## V. Case Study: Automobile Cruise Control over a CAN Bus

### A. System Model & Design Requirements

**System model.** Automotive cruise control tries to maintain a constant vehicle speed despite external disturbances, such as wind and road grade. It employs a PID controller [29] to automatically control the speed of a vehicle. It is a typical intermediate-value control problem [30].

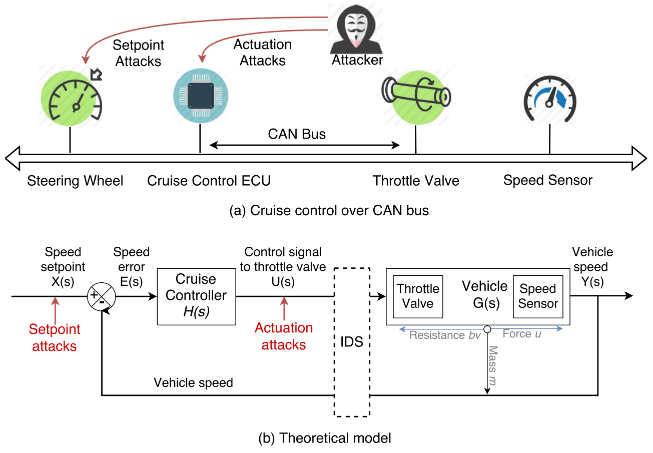

**Fig. 8:** System diagram of automobile cruise control.

Fig. 8(a) shows a simple automotive cruise control system [31] where messages are transmitted over a CAN (Controller Area Network) bus. The system contains a steering wheel, a cruise control ECU (Electronic Computing Unit), a throttle valve and speed sensors. The ECU controls the throttle valve through a vacuum actuator and takes sensing values from the sensors which monitor the vehicle speed and throttle position. Fig. 8(b) illustrates the theoretical model. A vehicle of mass $m$ is acted on by a control force $u$ adjusted by the throttle valve. The force $u$ is dampened by a resistance force of $bv$ in the opposite direction. The resistance force $bv$ represents disturbances such as the rolling resistance and the wind drag in the horizontal direction. We assume the control force $u$ can be directly controlled and thus we take $u$ as the control input. We neglect the dynamics of powertrain and tires that generate such a force.

Our cruise control system can be modeled by a first-order mass-damper system: $m\dot{v} + bv = u$, where $v$ is the vehicle speed, $\dot{v}$ is the acceleration, and $b$ is damping coefficient. The output value $y$ is chosen to be the speed of the vehicle: $y = v$.

The open-loop transfer function of the vehicle therefore is:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{ms + b} \qquad [\frac{m/s}{N}] \tag{9}$$

As an illustration, we choose the following parameters according to the example in [31]: vehicle mass $(m)$ = 1000 kg; damping coefficient $(b)$ = 50 N.s/m; nominal control force $(u)$ = 500 N.

**Intrusion detection systems.** A number of IDSs have been developed for the CAN bus to monitor and detect suspicious traffic inside a vehicle. Cho and Shin [32] leveraged clock skew as a fingerprint of different ECUs to detect fabrication, suspension and masquerade attacks. Due to the statistical nature of the underlying detection mechanisms, the online detection latency can reach several seconds. In this example, we assume a detection latency $t_p$ of 8.5 second.

**Design requirements.** In the open-loop step response, when the vehicle is given a step input of 500 Newtons force ($u = 500$), it will reach a maximum speed of 10 m/s (22 mph). The vehicle should be able to accelerate to that speed within $13s$ with a maximum overshoot of 10%. Moreover, the vehicle should not reach the setpoint within the IDS detection latency $t_p$. In short, the design requirements are the following: (1) $t_{max}$ = $13s$; (2) overshoot $< 10\%$; (3) $t_p$ = $8.5s$; and (4) $t_p \leq t_r \leq t_{max}$.

## B. Original System without CR

We use Ziegler-Nichols method to choose the initial PID parameters as $K_p = 289.74$, $K_i = 30.86$, and $K_d = 0$. The original PID controller is:

$$H(s) = \frac{K_p \cdot s + K_i + K_d \cdot s^2}{s} = \frac{289.74s + 30.86}{s} \quad (10)$$

The close-loop transfer function for this cruise control system is:

$$\begin{aligned}\frac{Y(s)}{X(s)} &= \frac{G(s) \cdot H(s)}{1 + G(s) \cdot H(s)} \\ &= \frac{K_p \cdot s + K_i + K_d \cdot s^2}{(m + K_d) \cdot s^2 + (K_p + b) \cdot s + K_i} \\ &= \frac{289.74s + 30.86}{1000s^2 + 339.74s + 30.86} \quad (11)\end{aligned}$$

Fig. 9 depicts the system step response of the original PID controller where the rise time is $8.0s$. With a single malicious setpoint message the attacker can drastically increase the vehicle speed (*e.g.* from 10 to 100 mph) within $8.0s$. Thus, the IDS on the CAN bus fails to detect suspicious packets in time because the detection latency is $t_p = 8.5s$.

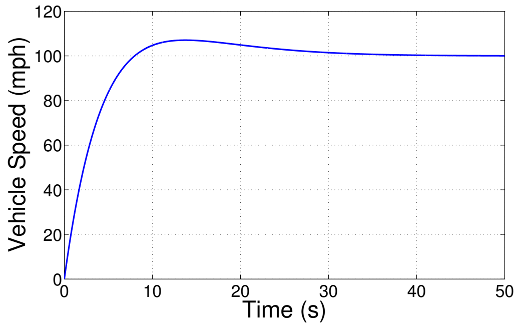

**Fig. 9:** Original cruise control step response with rise time of $8.0s$.

## C. System with Change-driven CR

Now, we insert change-driven CR module into the system. We adopt the aggressive mode CR where $C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{K_p \cdot s + K_i + K_d \cdot s^2}$. Thus, the new PID controller $H(s) \cdot C(s)$ turns out to be:

$$H(s) \cdot C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{s} \quad (12)$$

Given the plant dynamics $G(s)$, we run simulations to estimate the system response time $t_r$ with different PID parameters and follow the design steps in Section IV-B to create a scheduling policy, see Table II. We consider 9 ranges for $\Delta x$, each of which corresponds to a $t_r$ and a PID parameter tuple. The critical point of the system is 100 mph. Any speed below 100 mph is considered safe for people and the vehicle. Decreasing $K_p^{cr}$ and $K_i^{cr}$ may increase $t_r$.

| $\Delta x$ (mph) | $t_r$ | $K_p^{cr}$ | $K_i^{cr}$ | $K_d^{cr}$ |
|---|---|---|---|---|
| 0 - 20 | 8.5 | 273.09 | 28.30 | 0 |
| 20 - 30 | 9.0 | 258.20 | 26.09 | 0 |
| 30 - 40 | 9.5 | 244.81 | 24.18 | 0 |
| 40 - 50 | 10.0 | 233.58 | 22.62 | 0 |
| 50 - 60 | 10.5 | 222.63 | 21.15 | 0 |
| 60 - 70 | 11.0 | 212.67 | 19.84 | 0 |
| 70 - 80 | 11.5 | 203.56 | 18.68 | 0 |
| 80 - 90 | 12.0 | 195.43 | 17.66 | 0 |
| 90 - 100 | 12.5 | 188.28 | 16.79 | 0 |

**TABLE II:** Change-driven CR scheduling policy.

To illustrate the effect of change-driven CR, Fig. 10 shows the system step response with different rise time: $t_r \in [8.5, 12.5]$ with a step of 1 second. First, if attackers send a single command to increase the speed from 10 to 100 mph ($\Delta x$ = 90 mph), change-driven CR adopts PID parameters that result in $t_r = 12.5s$. The IDS will have enough time to detect and alerts will be issued at $4.0s$ when the vehicle speed is around 60 mph. Second, if attackers are aware of the CR module and want to avoid the additional response latency, they may choose to send 90 commands with 1 mph change in each message. This will generate a spike on the network traffic, which will also be easily identified by the IDS who monitors the CAN bus traffic. For commands with drastic changes in the setpoint, CR provides enough time for the IDS and system operators to respond or forces the attacker to take more effort (by sending more packets). Compared to the strategy where we increase $t_r$ to $12.5s$ regardless of $\Delta x$, change-driven CR makes the system more responsive for operations with smaller setpoint changes. Hence, change-driven CR greatly facilities the IDS to detect setpoint attacks and prevents drastic changes to take place within a short time.

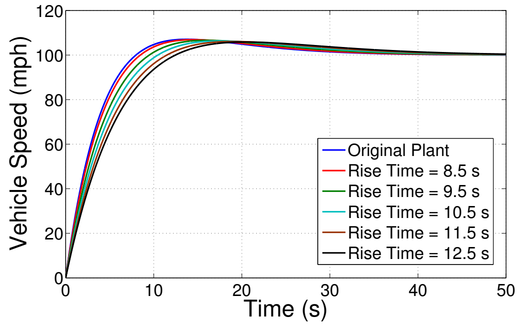

**Fig. 10:** Step response of cruise control with change-driven CR.

## D. System with Criticality-driven CR

| $\lvert y(t) - y_c \rvert$ (mph) | $t_r$ | $K_p^{cr}$ | $K_i^{cr}$ | $K_d^{cr}$ |
|---|---|---|---|---|
| 80 - 100 | 8.5 | 273.09 | 28.30 | 0 |
| 60 - 80 | 9.5 | 244.81 | 24.18 | 0 |
| 40 - 60 | 10.5 | 222.63 | 21.15 | 0 |
| 20 - 40 | 11.5 | 203.56 | 18.68 | 0 |
| 0 - 20 | 12.5 | 188.28 | 16.79 | 0 |

**TABLE III:** Criticality-driven CR scheduling policy.

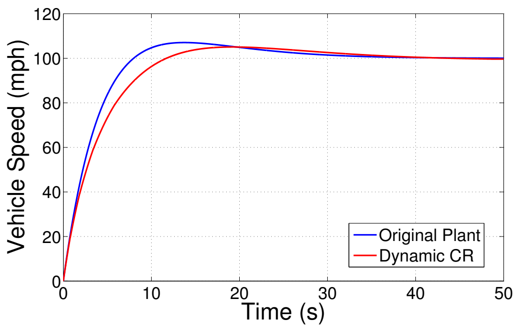

**Fig. 11:** Step response of cruise control with criticality-driven CR.

Sample criticality-driven CR scheduling policy is shown in Table III. The vehicle becomes slower in response as its speed $y(t)$ approaches the critical point $y_c$. Both the QoS requirements and the IDS timing requirements are satisfied. The impact of criticality-driven CR on the cruise control step response is shown in Fig. 11 (the vehicle speed increases from 0 to 100 mph). Criticality-driven CR (with the policy in Table III) increases $t_r$ from $8.0s$ (original system) to $11.71s$. Consider an attack scenario where the vehicle is operating at 70 mph, the attacker sends a false actuation signal to wide open throttle. Without criticality-driven CR, the car will get to 100 mph in less than $8.5s$. However if criticality-driven CR is in effect, $t_r$ will become $11.5s$ and IDS alerts will be issued at $3.0s$. If the attacker issues multiple malicious setpoint or actuation commands with smaller adjustments, the unusual traffic spike within a short time will also be obvious to the IDS. Both setpoint attacks and actuation attacks can lead to a change in $y(t)$, so criticality-driven CR can be used against both attack cases.

Change-driven CR and criticality-driven adjust the system from different perspectives: one focuses on the degree of change while the other focuses on the criticality of the current operating state. In practical cases, both strategies can be combined together to better harden the security protection of a CPS. For example, if the vehicle is operating at 85 mph ($|y(t) - y_c|$ = 15 mph) and the attacker issues a false command with the setpoint of 100 mph ($\Delta x$ = 15 mph), criticality-driven CR will dominate the CR module because it considers a more dangerous situation with $t_r$ = $12.5s$ (versus $t_r$ = $8.5s$ suggested by change-driven CR).

## VI. Discussion

CR prevents malicious attacks by slowing down on-demand the system response to sudden changes. When the desired change is not from an attack, this slowing down increases the latency for the system to reach the desired state. In such cases, the additional response latency is the cost of our approach. We note that this occurs rarely, in particular for criticality-driven CR, because the system response time during normal operation is left largely unchanged.

We currently adopt a greedy algorithm (the smallest norm-2 distance to the last selected PID parameter tuple) to pick a PID parameter tuple from a set of results that satisfy the design requirements of the original system. No new design requirements (such as QoS) need to be addressed. We could also choose to optimize one or more specific design requirements, such as minimizing rise time, settling time, and overshoot) and select the PID parameters accordingly.

As alternative to CR one may want to consider network-level IDS approaches. For example, DPI (deep packet inspection) could be used to identify setpoint attacks by inspecting the application-layer data payload. Overly aggressive setpoint-change commands would be simply rejected by the IDS, thus providing a seemingly simple protection scheme against such attacks. Unfortunately, intelligent attackers would tailor their commands and send sequences of small-change commands. The IDS would then have to rate-control these commands to protect the plant, which is fundamentally no different than CR. In addition, criticality-driven CR can prevent stealthy attacks that cannot be detected by the DPI, *e.g.* malicious setpoint commands with smaller setpoint changes sent over a longer period of time. In addition to being at least as effective as DPI, CR provides an integrated analysis framework that allows to safely determine the parameters of the CR modules to protect the system while maintaining its operational requirements.

## VII. Related Work

Security issues of CPS can be tackled from two perspectives: communication network approaches and system-theoretic approaches [33]. Both are critical to secure CPS: communication network approaches can effectively prevent or thwart cyber-attacks from getting into a CPS while system-theoretic solutions rely on physical plant dynamics and system states to identify security-important sensor nodes and malicious behavior.

**Communication networks approaches.** Communication networks approaches enhance security protections from SCADA networks. Protection mechanisms have been proposed to secure critical infrastructures from intrusion detection systems (IDS) [6]–[10], [34], path redundancy [12], [13], authentication [11], [14] and encryption [15], [35], etc.

**System-theoretic approaches.** System-theoretic approaches assume attackers have already compromised some physical components or malicious packets have already been inserted into the system. These solutions rely on the physical process dynamics to detect different kinds of cyber and physical attacks. However, the models are constructed based on approximations and subject to noise, which can result in a deviation of any model to the reality.

Several integrity attacks, including false data injection attacks (FDIA), reply attacks, stealthy attacks, and DoS attacks have been discussed in [36], [37]. FDIA [19], [22], which target against state-estimation and can circumvent the bad data detection mechanisms in the electrical grid, have been well studied in the context of smart grid. Liu et. al [19] showed how FDIA can be staged; Xie et. al [20] showed that FDIA can lead to profitable financial misconduct in electrical market; and Kwon et. al [22] focus on three kinds of stealthy attacks and provide conditions under which attacks would succeed without being detected.

Replay attacks have also been discussed in [23]. The authors analyze attack models inspired by Stuxnet. Several defense mechanisms have been proposed, among which dynamic watermarking [16]–[18] can actively detect such attacks. It tries to identify malicious sensors by injecting small excitation into the system and detecting ones that report false sensing values. While a number of approaches [18], [19], [23] develop statistical hypothesis tests that detect attacks with a certain error rate, dynamic watermarking [16], [17] ensures that the amount of distortion that attackers can add without exposing their presence can have an average power of only zero.

## VIII. Conclusion

Cyber-attack incidents against cyber-physical systems have shown that an adversary can make drastic changes to a plant within a short time. To deal with this problem, we introduce adaptive Commensurate Response (CR) as a cross-domain technique that protects a control system through appropriate control parameter selection against attacks coming from the communication domain. We provide detailed study on both change-driven CR and criticality-driven CR. Our case study on automobile cruise control demonstrates that change-driven CR can be used against setpoint attacks and criticality-driven CR is effective to combat both setpoint and actuation attacks. CR can effectively improve the system resilience and attack survivability and facilitate other defense mechanisms such as firewalls and IDS to better protect CPS. In future work, we plan to combine CR with additional QoS constraints and consider other types of controllers.

## IX. Acknowledgements

This research is supported by a grant from Qatar National Research Foundation 9th Cycle under Grant no. 9-069-1-018 and a seed grant from Texas A&M Engineering Experiment Station (TEES).

## References

[1] X. Yu and Y. Xue, "Smart grids: A cyber–physical systems perspective," *Proc. of the IEEE*, vol. 104, no. 5, pp. 1058–1070, 2016.

[2] Wikipedia, "Building automation — wikipedia, the free encyclopedia," 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Building_automation&oldid=776185496

[3] ——, "Sewage treatment — wikipedia, the free encyclopedia," 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Sewage_treatment&oldid=777151381

[4] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[5] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," *SANS Indus. Control Sys.*, 2016.

[6] H. Lin, A. Slagell, C. Di Martino, Z. Kalbarczyk, and R. K. Iyer, "Adapting bro into scada: building a specification-based intru. det. syst. for the dnp3 protocol," in *CSIIRW'13*. ACM, 2013, p. 5.

[7] J. Verba and M. Milvich, "Idaho natl. lab. supry. ctrl. and data acq. intru. detct. syst. (scada ids)," in *HST'08*. IEEE, 2008, pp. 469–473.

[8] R. R. R. Barbosa and A. Pras, "Intrusion detection in scada networks," in *AIMS'10*. Springer, 2010, pp. 163–166.

[9] J. Bigham, D. Gamez, and N. Lu, "Safeguarding scada sys. with anomaly detection," in *Intl. Wksh. on Math. Methods, Models, and Arch. for Compt. Ntwk. Sec.* Springer, 2003, pp. 171–182.

[10] N. Goldenberg and A. Wool, "Accurate modeling of modbus/tcp for intrusion detection in scada systems," *IJCIP*, vol. 6, 2013.

[11] C. R. Taylor, C. A. Shue, and N. R. Paul, "A depl. scada auth. tech. for modern power grids," in *ENERGYCON*. IEEE, 2014.

[12] Z. Zheng and A. L. N. Reddy, "Towards improving data validity of cyber-physical systems through path redundancy," in *ACM CPSS'17*. ACM, 2017, pp. 91–102.

[13] D. Germanus, A. Khelil, and N. Suri, "Increasing the resilience of critical scada systems using peer-to-peer overlays," in *Architecting Critical Systems*. Springer, 2010, pp. 161–178.

[14] B. Vaidya, D. Makrakis, and H. T. Mouftah, "Authentication and authorization mechanisms for substation automation in smart grid network," *IEEE Network*, vol. 27, no. 1, pp. 5–11, 2013.

[15] A. K. Wright, J. A. Kinast, and J. McCarty, "Low-latency crypto. protection for scada comm." in *ANCS'04*. Springer, 2004.

[16] B. Satchidanandan and P. Kumar, "Dynamic watermarking: Active defense of netwk. cyber-physical systems," *Proc. of the IEEE*, 2016.

[17] W.-H. Ko, B. Satchidanandan, and P. Kumar, "Theory and impl. of dynamic watermarking for cybersecurity of advanced trans. syst." in *IEEE CNS'16*. IEEE, 2016, pp. 416–420.

[18] S. Weerakkody, Y. Mo, and B. Sinopoli, "Detecting integrity attacks on control systems using robust physical watermarking," in *IEEE CDC'14*. IEEE, 2014, pp. 3757–3764.

[19] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *TISSEC*, vol. 14, no. 1, 2011.

[20] L. Xie, Y. Mo, and B. Sinopoli, "False data injection attacks in electricity markets," in *IEEE Intl. Conf. on Smart Grid Comm. (SmartGridComm'10)*. IEEE, 2010, pp. 226–231.

[21] G. Hug and J. A. Giampapa, "Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks," *IEEE Trans. on Smart Grid*, vol. 3, no. 3, pp. 1362–1370, 2012.

[22] C. Kwon, W. Liu, and I. Hwang, "Security analysis for cyber-physical systems against stealthy deception attacks," in *American Control Conference (ACC'13)*. IEEE, 2013, pp. 3344–3349.

[23] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *47th Ann. Allerton Conf. on Comm. Control, and Compt. (Allerton'09)*. IEEE, 2009, pp. 911–918.

[24] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *Intl. Conf. on Dist. Compt. Syst. Workshops (ICDCS'08)*. IEEE, 2008, pp. 495–500.

[25] A. Manocha and A. Sharma, "Three axis aircraft autopilot control using genetic algorithms: An experimental study," in *IEEE Intl. Advance Compt. Conf. (IACC'09)*. IEEE, 2009, pp. 171–174.

[26] J. Ziegler and N. Nichols, "Optimum settings for automatic controllers," *ASME DC*, vol. 115, no. 2B, pp. 220–222, 1993.

[27] M. L. Luyben and W. L. Luyben, *Esse. of proc. ctrl.* McGraw-Hill, 1997.

[28] S. V. Labs, "Designing of reconfigurable architecture for pid controller," 2017. [Online]. Available: http://coep.vlab.co.in/index.php?sub=29&brch=88&sim=1312&cnt=1

[29] P. Ioannou, Z. Xu, S. Eckert, D. Clemons, and T. Sieja, "Intel. cruise control: theory and expt." in *CDC'93*. IEEE, 1993, pp. 1885–1890.

[30] R. Rajamani, *Vehicle dyn. and control*. Sci. & Bus. Media, 2011.

[31] D.-T. HENTUNEN, "Control tutorials for matlab & simulink," June 2014, http://ctms.engin.umich.edu/CTMS/index.php?aux=Home.

[32] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Sec. Symp. (USENIX Security'16)*. USENIX Association, 2016, pp. 911–927.

[33] Y. Mo, T. H.-J. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli, "Cyber–physical security of a smart grid infrastructure," *Proc. of the IEEE*, vol. 100, no. 1, pp. 195–209, 2012.

[34] Z. Zheng and A. L. N. Reddy, "Safeguarding building automation networks: The-driven anomaly detector based on traffic analysis," in *IEEE ICCCN*. IEEE, 2017.

[35] S. L. Keoh, K. W. K. Au, and Z. Tang, "Securing indus. ctrl. sys.: An end-to-end integ. verif. appr." in *ICSS'15*. ACSA, 2015.

[36] Y. Mo and B. Sinopoli, "Integrity attacks on cyber-physical systems," in *Proc. High Conf. Ntwk. Sys.* ACM, 2012, pp. 47–54.

[37] M. Krotofil and A. A. Cárdenas, "Resil. of proc. ctrl. syst. to cyber-phy. attacks," in *Nordic Conf. on Sec. IT Sys.* Springer, 2013.