Adaptive Window-Based Sensor Attack Detection for Cyber-Physical Systems

Lin Zhang, Zifan Wang, Mengyu Liu, and Fanxin Kong
Department of Electrical Engineering and Computer Science, Syracuse University
{lzhan120,zwang345,mliu71,fkong03}@syr.edu

ABSTRACT

Sensor attacks alter sensor readings and spoof Cyber-Physical Systems (CPS) to perform dangerous actions. Existing detection works tend to minimize the detection delay and false alarms at the same time, while there is a clear trade-off between the two metrics. Instead, we argue that attack detection should dynamically balance the two metrics when a physical system is at different states. Along with this argument, we propose an adaptive sensor attack detection system that consists of three components - an adaptive detector, detection deadline estimator, and data logger. It can adapt the detection delay and thus false alarms at run time to meet a varying detection deadline and improve usability (or false alarms). Finally, we implement our detection system and validate it using multiple CPS simulators and a reduced-scale autonomous vehicle testbed.

KEYWORDS

cyber-physical systems, attack detection, detection deadline

ACM Reference Format:
Lin Zhang, Zifan Wang, Mengyu Liu, and Fanxin Kong. 2022. Adaptive Window-Based Sensor Attack Detection for Cyber-Physical Systems. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC) (DAC '22), July 10–14, 2022, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3489517.3530555

1 INTRODUCTION

Cyber-Physical Systems (CPS) integrate computing and networking components with physical processes through sensing and actuator units. One crucial security risk in CPS is sensor attacks, where an attacker alters sensor measurements to negatively interfere with the physical system. When acting on malicious sensor information, a controller can drive the physical system to unsafe states. In addition to software and network attacks, transduction attacks have been emerging to non-invasively affect sensor readings through manipulating a physical property [6, 12, 15]. For example, an attacker can inject fake GPS signals to guide a yacht off course [7] or use sound noise to affect gyroscope readings [9].

These new threats have motivated many research efforts on sensor attack detection. Usually, they detect attacks by identifying anomalies between observed sensor measurements and predicted values [1, 3, 6, 11]. Existing works attempt to minimize the detection delay and maximize usability at the same time. The detection delay is the time between the onset of an attack and the detection of it. Usability refers to the false alarm rate, i.e., a lower rate means better usability. It is, however, difficult to achieve the above goal due to the clear trade-off between the two metrics, i.e., greater usability usually comes with a longer detection delay [2, 10]. Instead, we propose that attack detection should bias the different metrics when a CPS runs in different states. For instance, if the current state of a physical system is close to the unsafe region, lowering the detection delay is preferable over reducing false alarms; and vice versa.

However, it is challenging to realize this adaptability in attack detection. First, timing is essential for safety-critical CPS, i.e., detection of an attack after consequences occur, is just as damaging. For example, detecting an attack after car accidents is useless. This time constraint is known as the detection deadline, before which attacks must be detected. Thus, the detection delay should not exceed the deadline. Second, it is not trivial to online calculate a detection deadline, which varies as the physical system state evolves. The overhead of the calculation should be low; otherwise, the calculated deadline may be outdated. In addition, a detector with fixed or unpredictable detection delay is inapplicable to match the varying deadline. Third, the detection delay should not be greater than the deadline, but a shorter detection delay is not always favorable. For example, a detector that raises an alert at every control period can discover every attack once it occurs, i.e., it has the shortest detection delay. However, the detector will raise an unmanageable number of false alarms and thus unacceptable usability. On the contrary, to determine the occurrence of an attack, a detector can raise an alert after monitoring multiple control periods. However, this will increase the detection delay.

To address these challenges, we propose a real-time adaptive attack detection system that can dynamically adjust detection delay and thus false alarms according to the varying system state. Our detection system will have a shorter detection delay and more false alarms when the detection deadline is stringent; and vice versa. Our system has three major components (shaded box in Fig. 1), and the technical contribution for each component is as follows.

•*Detection Deadline Estimator.* We develop a reachability-based technique to conservatively estimate the detection deadline on the fly. The technique finds the vulnerability of a system by computing the reachable set of future potential behaviors of the system.

•*Adaptive Detector.* We develop a window-based detection algorithm that can dynamically adapt its detection delay according to the deadline without missing any data points during the adaptation.

•*Data Logger.* We develop a sliding-window based data logging protocol, which keeps sufficient trustworthy data for deadline estimation and attack detection even when the detection delay varies.

DAC '22, July 10–14, 2022, San Francisco, CA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530555

We implement our detection system and validate it using multiple CPS simulators and a reduced-scale autonomous vehicle testbed. The results show the efficiency and efficacy of our detection system.

2 PRELIMINARIES AND DESIGN OVERVIEW

This section introduces the system and threat model and presents an overview of our adaptive sensor attack detection system.

System Model. We consider a CPS model where a physical system is controlled by a computer program or controller. At each t^{th} control step, sensors measure the state of the physical system and send them to the controller. Based on the sensor measurements, the controller obtains the state estimate $\bar{\boldsymbol{x}}_t$ of the physical system and generates control inputs $\boldsymbol{u}_t$ according to its control algorithm. The control inputs are then sent to actuators who apply $\boldsymbol{u}_t$ to supervise the physical system at a desired (or reference) state. The state of a physical system or the system state $\boldsymbol{x}_t \in \mathcal{X}$ is a vector of size n that represents the number of dimensions of the system state (e.g., velocity, electric current, pressure, etc.). The control inputs $\boldsymbol{u}_t \in \mathcal{U}$ is a vector of size m that represents the number of dimensions of the control input (e.g. steering angle, applied voltage, etc.). Note that $\mathcal{U}$ is the control input range that is usually limited by the actuator's capability, and for instance, the acceleration of a vehicle is limited by the capacity of the accelerator.

The dynamics of a physical system obey physics laws and can be modeled by a set of differential or difference equations. In this paper, we consider a discrete linear time-invariant (LTI) model,

$$\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + \boldsymbol{v}_t, \tag{1}$$

where $\boldsymbol{v}_t$ is uncertainty (such as modeling error and disturbance), and matrices A and B are state and input matrix with suitable dimensions, indicating how future states evolve from the current state and control input. For ease of presentation, we assume that the physical system is fully observable (i.e., all n dimensions can be estimated from sensor measurements).

Threat Model. The system state is estimated from sensor measurements. Thus, an attacker can manipulate sensor measurements to compromise state estimates. The difference between state estimate $\bar{\boldsymbol{x}}_t$ and system state $\boldsymbol{x}_t$ denoted by vector $\boldsymbol{e}_t \in \mathbb{R}^n$. The state estimate can be partially or fully affected, i.e., the number of non-zero dimensions of $\boldsymbol{e}_t$ or l_0 norm of $\boldsymbol{e}_t$ is $0 < ||\boldsymbol{e}_t||_0 < n$, or $||\boldsymbol{e}_t||_0 = n$. An attacker can compromise state estimates by corrupting the integrity (e.g. transduction attacks) or availability (e.g. DoS attacks) of sensors. These attacks result in misleading control inputs that drive the physical system to undesired states and even cause serious consequences.

Overview of the Attack-Detection Framework. Our adaptive sensor attack detection framework is illustrated in Fig. 1. It consists of three components in the shaded box: (1) *Adaptive Detector*, (2) *Detection Deadline Estimator*, and (3) *Data Logger*. The following briefly introduces these components and their detailed design will be presented in Sections 3, 5 and 4, respectively.

First, the Detection Deadline Estimator conservatively calculates the detection deadline, after which the physical system may reach unsafe states. Thus, the attack detector should find attacks before the deadline. Note that the detection deadline may vary over time as the system state changes and thus the Adaptive Detector needs to adapt the detection delay accordingly. Second, the Data Logger logs

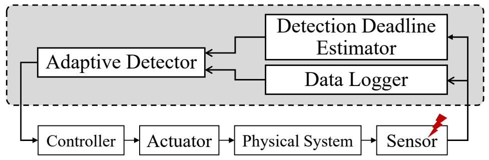


Figure 1: Design overview of the real-time adaptive sensor attack detection system.

state estimates and residuals. At every control period, it predicts expected state and calculates the residual (or difference) between the predicted value and the observed value. It records enough data points for the Adaptive Detector to calculate the cumulative residual, even when the detection delay varies. Third, based on their outputs, the Adaptive Detector will track the cumulative residual. When the average residual in the detection window becomes larger than a predefined threshold, the detector will raise an alarm to signal the detection of an attack. Importantly, the detector can dynamically adapt its detection delay to meet the detection deadline.

3 DETECTION DEADLINE ESTIMATION

This section presents the design of Detection Deadline Estimator. We use reachability analysis to conservatively estimate the detection deadline. The following first defines the safety analysis problem, then presents how to approximate the reachable set, and finally shows how to calculate the deadline using the support function.

3.1 Safety Analysis

When applying a sequence of control inputs $\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{T-1} \in \mathcal{U}$ to a physical system, the system state evolves according to its dynamics ψ, i.e., $\boldsymbol{x}_{t+1} = \psi(\boldsymbol{x}_t, \boldsymbol{u}_t)$. The sequence of evolving states is called *State Trajectory* ξ, where ξ_i denotes the i-th state in the trajectory. By applying all possible control sequences within T control steps, we can have all possible state trajectories $\Xi(\boldsymbol{x}_0, T)$ as $\Xi(\boldsymbol{x}_0, T) = \{\xi : \xi_0 = \boldsymbol{x}_0, \xi_{t+1} = \psi(\xi_t, \boldsymbol{u}_t)\}$, where $\boldsymbol{u}_t \in \mathcal{U}$, $t \in \{0, \ldots, T-1\}$, and $\boldsymbol{x}_0$ is an initial state. Then, the reachable set $\mathcal{R}$ includes all possible system states in $\Xi(\boldsymbol{x}_0, T)$.

The *Unsafe State Set* $\mathcal{F}$ is a region within the state space, in which the physical system is unsafe and may cause serious consequences. For example, the distance from a vehicle to an obstacle is less than zero where the unsafe state includes all negative distance values. The complementary set of $\mathcal{F}$ is the safe state set $\mathcal{S}$. To keep the system safe, the reachable set of a system from a certain initial state $\boldsymbol{x}_0$ is required not to intersect with the unsafe state set, i.e., $\mathcal{R} \cap \mathcal{F} = \emptyset$. Unfortunately, it is very expensive to compute the exact reachable set. Instead, we usually compute an over-approximation of the reachable set, denoted by $\bar{\mathcal{R}}$ and $\bar{\mathcal{R}} \supseteq \mathcal{R}$. If $\bar{\mathcal{R}} \cap \mathcal{F} = \emptyset$, then we can guarantee that $\mathcal{R} \cap \mathcal{F} = \emptyset$. Based on this, we define conservatively safe as Definition 3.1.

Definition 3.1. *The system is Conservatively Safe, if the over-approximation of the reachable set does not intersect with the unsafe state set* $\mathcal{F}$, *i.e.,* $\bar{\mathcal{R}} \cap \mathcal{F} = \emptyset$.

3.2 Over-approximation of the Reachable Set

Given the system model by Eq. (1), a state trajectory is affected by both uncertainty and control input. To calculate the reachable set, we need to over-approximate both parts [5]. The over-approximation uses the ball and box defined as follows.

Definition 3.2. *A unit ball is the closed set of points whose k-norm distance is less than or equal to 1 from a fixed central point. For n dimensions and any $k > 1$, the origin-centered unit ball $\mathcal{B}_{(k)}$ is defined as $\mathcal{B}_{(k)} = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_k = (\sum_{i=0}^{n-1} |x_{(i)}|^k)^{\frac{1}{k}} \le 1\}$.*

Definition 3.3. *A box is a set that can be denoted by a product of intervals, i.e., $[x^l_{(1)}, x^u_{(1)}] \times \cdots \times [x^l_{(n)}, x^u_{(n)}]$, where $x^l_{(i)}$ and $x^u_{(i)}$ are the lower and upper bounds of $\boldsymbol{x}$'s the i-th dimension.*

Especially, any 2-norm ball (*Euclidean ball*) can be scaled from a unit Euclidean ball. The infinity-norm unit ball $\mathcal{B}_{(\infty)}$ is a box, i.e., $\mathcal{B}_{(\infty)} = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_\infty = \max_{0 \le i < n} |x_{(i)}| \le 1\}$, where $x_{(i)}$ is the i-th dimension of $\boldsymbol{x}$. Hence, any box can be transformed from an infinity-norm unit ball by scaling in each dimension.

3.2.1 *Over-approximation of uncertainty.*

We assume that $\boldsymbol{v}_t$ in Eq. (1) by is bounded an error $\epsilon > 0$ at one control step . Thus, it can be over-approximated by an origin-centered euclidean ball $\mathcal{B}_\epsilon$ with radius ϵ. That is, we have $\boldsymbol{x}_t \in \tilde{\boldsymbol{x}}_t \oplus \mathcal{B}_\epsilon$, where $\tilde{\boldsymbol{x}}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{u}_{t-1}$ and $\oplus$ denotes the Minkowski sum. The Minkowski sum is defined as $X \oplus Y = \{x + y | x \in X, y \in Y\}$ for any set X and Y.

3.2.2 *Over-approximation of control input set.*

Consider the control input set $\mathcal{U} = [u_{(1)}, ..., u_{(m)}]$. For the i-th dimension $u_{(i)}$, it has the upper and lower bounds denoted as $u^u_{(i)}$ and $u^l_{(i)}$, respectively. Then, the control input set can be over-approximated by a box $\mathcal{B}_\mathcal{U}$ with a center $\boldsymbol{c}$, where $c_{(i)} = (u^u_{(i)} + u^l_{(i)})/2$. This box can be scaled from a unit infinity-norm ball $\mathcal{B}_{(\infty)}$ with a scaling factor γ_i in the i-th dimension, where $\gamma_i = (u^u_{(i)} - u^l_{(i)})/2$. The box is expressed by the transformation of the infinity-norm unit ball, given by $\mathcal{B}_\mathcal{U} = \boldsymbol{c} + Q\mathcal{B}_{(\infty)}$, where $Q = \mathrm{diag}(\gamma_1, \cdots, \gamma_m)$, i.e., a $m \times m$ matrix with $\{\gamma_1, \cdots, \gamma_m\}$ in its diagonal.

Note that the uncertainty $\boldsymbol{v}_t$ can be over-approximated by a Euclidean ball by nature. In CPS, each actuator has its own control input range or interval. Thus, the control input set can be expressed by a product of these intervals, which is then a box.

3.2.3 *Over-approximation of the Reachable set.*

Given the over-approximated control input set $\mathcal{B}_\mathcal{U}$, the over-approximated uncertainty $\mathcal{B}_\epsilon$, and an initial state $\boldsymbol{x}_0$ according to the system model by Eq. (1), the system state $\boldsymbol{x}_t$ will be bounded by the over-approximation of reachable set $\bar{\mathcal{R}}(\boldsymbol{x}_0, t)$, given by Eq. (2).

$$\boldsymbol{x}_t \subseteq A^i \bar{\boldsymbol{x}}_0 \oplus \bigoplus_{j=0}^{i-1} A^j B \mathcal{B}_\mathcal{U} \oplus \bigoplus_{k=0}^{i} A^k \mathcal{B}_\epsilon \tag{2}$$

3.3 Deadline Searching Process

3.3.1 *Selection on the initial state $\boldsymbol{x}_0$.*

First, we calculate the reachable set from the latest trustworthy state estimation $\bar{\boldsymbol{x}}_{t-w_c-1}$ that has just moved outside the detection window, i.e., $\boldsymbol{x}_0 = \bar{\boldsymbol{x}}_{t-w_c-1}$. It correctly reflects the physical state at that time, while state estimates within the detection window, i.e., $\{\bar{\boldsymbol{x}}_{t-w_c}, ..., \bar{\boldsymbol{x}}_t\}$, are still questionable. The time point is $t - w_c - 1$, where t and w_c are the current time and window size, as illustrated in Fig. 5. (More details on data logging are in Section 5.) Second, if we consider some noise in state estimates, we can use an initial state set containing $\boldsymbol{x}_0$. The initial set can be bounded by a ball given bounded noise. Then, we can apply the same reachability analysis as above, too.

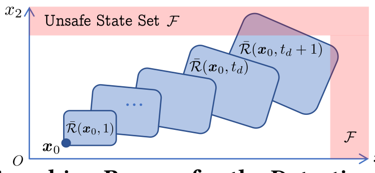

x_2
Unsafe State Set $\mathcal{F}$
$\bar{\mathcal{R}}(\boldsymbol{x}_0, t_d + 1)$
$\bar{\mathcal{R}}(\boldsymbol{x}_0, t_d)$
$\cdots$
$\bar{\mathcal{R}}(\boldsymbol{x}_0, 1)$
$\boldsymbol{x}_0$
$\mathcal{F}$
O
x_1

Figure 2: Searching Process for the Detection Deadline t_d.

3.3.2 *Deadline searching process.*

Starting from $\boldsymbol{x}_0$, we compute the reachable set at each subsequent step until there is an intersection between the over-approximation of reachable set $\bar{\mathcal{R}}(\boldsymbol{x}_0, t)$ and the unsafe state set $\mathcal{F}$, or it exceeds the maximum deadline given beforehand (i.e, the maximum detection window size as in Section 4.3). Let say we find the first intersection at the $(t_d + 1)$-th step, the system is safe before this step according to the Definition 3.1, and thus t_d is regarded as the detection deadline. The detector is expected to identify an attack within the deadline.

Fig. 2 depicts an example that illustrates the detection deadline search process for the system state with two dimensions of x_1 and x_2. At t_d-th step, $\bar{\mathcal{R}}(\boldsymbol{x}_0, t_d) \cap \mathcal{F} = \emptyset$, and at $(t_d + 1)$-th step, $\bar{\mathcal{R}}(\boldsymbol{x}_0, t_d + 1) \cap \mathcal{F} \neq \emptyset$. Hence, the detection deadline is set to t_d.

3.4 Computing deadline using support function

According to Eq. (2), we can see that the over-approximation is difficult to compute because of the operation of the Minkowski sum. We choose to use the support function method [5] to derive a box over-approximation of the reachable set $\bar{\mathcal{R}}(\boldsymbol{x}_0, t)$. For a vector $\boldsymbol{l}$, the *support function* of a set $\mathcal{S} \subseteq \mathbb{R}^n$ is defined as $\rho_\mathcal{S}(\boldsymbol{l}) = \sup_{\boldsymbol{x} \in \mathcal{S}} (\boldsymbol{l}^T \boldsymbol{x})$. Then, according to the properties of support function, we can derive the support function of the reachable set $\bar{\mathcal{R}}(\boldsymbol{x}_0, t)$ using Eq. (3).

$$\rho_{\bar{\mathcal{R}}} = \boldsymbol{l}^T (A^t \boldsymbol{x}_0) + \sum_{i=0}^{t-1} \rho_{\mathcal{B}_\mathcal{U}}((A^i B)^T \boldsymbol{l}) + \sum_{i=0}^{t-1} \rho_{A^i \mathcal{B}_\epsilon}(\boldsymbol{l}) \tag{3}$$

Based on Eq. (3), we can have the upper and lower bounds of $\bar{\mathcal{R}}(\boldsymbol{x}_0, t)$ in the i-th dimension, as shown in Eq. (4) and (5), respectively, where $\boldsymbol{l}$ is set to be a column vector whose i-th entry is 1 and the others are 0.

$$\boldsymbol{l}^T A^t \boldsymbol{x}_0 + \sum_{i=0}^{t-1} \boldsymbol{l}^T A^i B \boldsymbol{c} + \sum_{i=0}^{t-1} \|(A^i B Q)^T \boldsymbol{l}\|_1 + \sum_{i=0}^{t-1} \epsilon \|(A^i)^T \boldsymbol{l}\|_2 \tag{4}$$

$$\boldsymbol{l}^T A^t \boldsymbol{x}_0 + \sum_{i=0}^{t-1} \boldsymbol{l}^T A^i B \boldsymbol{c} - \sum_{i=0}^{t-1} \|(A^i B Q)^T \boldsymbol{l}\|_1 - \sum_{i=0}^{t-1} \epsilon \|(A^i)^T \boldsymbol{l}\|_2 \tag{5}$$

Finally, by comparing the upper and lower bounds with the unsafe state set (i.e., a similar search process as in Fig. 2), we can know when the reachable set has an intersection with the unsafe set, and thus determine the detection deadline.

4 ADAPTIVE WINDOW BASED ATTACK DETECTION

This section presents the design of Adaptive Detector. We enhance window-based detection to accommodate varying window sizes. The following first gives the basic window-based detection and then proposes protocols on adapting the window size or the detection delay according to the deadline from Detection Deadline Estimator.

4.1 Basic Window Based Detection

This basic detection algorithm tracks the residual at each control step. The residual $\boldsymbol{z}_t$ is defined as the difference between the predicted state $\tilde{\boldsymbol{x}}_t$ and the state estimate $\bar{\boldsymbol{x}}_t$, where $\tilde{\boldsymbol{x}}_t = A\bar{\boldsymbol{x}}_{t-1} + B\boldsymbol{u}_{t-1}$. Residuals will be provided by the Data Logger.

First, the algorithm will calculate the average residual in the detection window $\boldsymbol{z}^{avg}_t = \frac{1}{w_c} \sum_{i \in [t-w_c, t]} |\tilde{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_i|$, where t is the

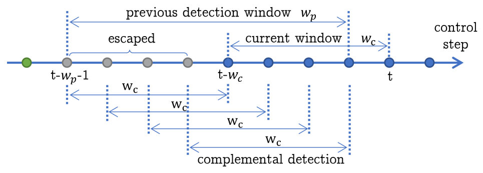


Figure 3: Decreasing the Detection Window Size

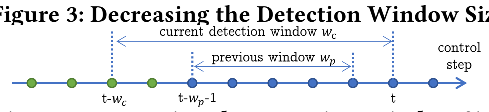


Figure 4: Increasing the Detection Window Size

current time and w_c is the current detection window size. Then, the algorithm will compare z_t^{avg} with a threshold τ. If $z_t^{avg} \leq \tau$, no alarm will be raised and the system is regarded as secure. Otherwise, an alarm will be raised to signal the detection of an attack.

Note that there are two hyper-parameters in this algorithm - the threshold τ and the detection window size. Because we focus on the timing aspect, dynamically adjusting the threshold is not the focus of this paper. The focus is to adjust the detection window size on the fly. To understand the rationale behind this, we need to elucidate the relationship between the window size and the detection delay. Since data points that lay outside the detection window are treated uncompromised, attacked data points are inside the window. Thus, the window size bounds the detection delay, i.e., the maximum detection delay is the window size. Further, with a longer detection delay, a detector tends to have lower false alarm rates but may miss the detection deadline; and vice versa. This is clearly observed in our experimental results in Section 6.

4.2 Detection Window Adjustment Protocol

Based on the rationale above, our protocol sets the window size as the detection deadline, online calculated by Detection Deadline Estimator. If the deadline decreases, the protocol will shrink the detection window to meet the timing constraint; otherwise, the protocol will enlarge the detection window to reduce false alarms.

4.2.1 Decreasing the window size. Fig. 3 shows the case that the detection window is decreasing. The size of the previous detection window is indicated as w_p. The green dots represent the state estimates that move outside the previous window, and their detection results are finalized and trusted to be uncompromised. On the contrary, state estimates within the detection window may be attacked, but not been detected yet. We need to ensure that these data points do not escape detection when reducing the window size. Thus, the protocol performs the following steps.

At the current time t, the detection window becomes w_c, where $w_p > w_c$. First, we set $w_c = t_d$, and t_d is the detection deadline at the current time. Note that the data (marked in grey) within previous detection window but outside current window (i.e., from $t - w_p - 1$ to $t - w_c - 1$) are escaped from the current shorter detection window. Thus, before the detection for current control step t, the complemental detection runs the detection algorithm with window size w_c from control step $t - w_p - 1 + w_c$ to $t - 1$, as shown in Fig. 3. By doing this, there will be no data that can escape from the current shorter detection window without checking.

4.2.2 Increasing the window size. Fig. 4 illustrates the case that the detection window size is increasing, where the current detection window w_c is larger than the previous detection window w_p. State estimates marked in green have moved outside the previous detection window, and thus their detection results are finalized. Note that part of these state estimates re-enter the detection window at the current time, but no data points escape from the current longer detection window w_c. Thus, we can continue the window-based detection algorithm with a longer detection window directly, and complemental detection is not needed in this case.

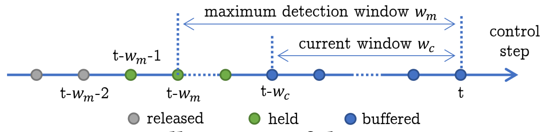


Figure 5: Illustration of the Data Logger.

4.3 The Maximum Window Size and False Negatives

We predefine a maximum detection window size w_m. At run time, the window size will be adjusted in the range of $[0, w_m]$. If the detection deadline t_d is greater than the maximum window size w_m, then the window size will be set as the maximum, i.e., $w_c = w_m$. Note that as mentioned, this maximum size is also the termination condition for the deadline searching process if no intersection with the unsafe set is found in the first w_m steps.

To decide an appropriate maximum detection window size, we perform offline profiling. The profiling establishes a relationship between false negatives/positives and the window size. We will experiment with a long enough range of window size, and cut out the sub-range with an acceptable false negative rate. This cutting line can be given by a specific application. For example, as shown in Fig. 7, to avoid false negative experiments (attacks are not detected), the maximum window size can be set as 35 control steps; to tolerate 3 false negative experiments out of 100, the maximum window size can be set as 40 control steps. More details are presented in Section 6.

Also, note that adjusting the detection window size makes sense only if attacks can be detected by the window-based detection algorithm. For false negatives, regulating the threshold τ in Section 4.1 is more desired, but this is not the focus of this paper. Instead, this paper focuses on the timing aspect, i.e., the detection delay.

5 DATA LOGGING PROTOCOL

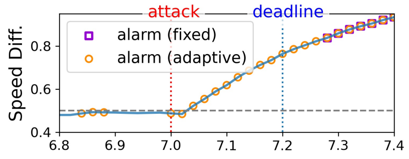
This section presents the design of the Data Logger. We adapt a sliding-window-based logging protocol to record historical residuals and state estimates [13]. To keep sufficient data for the other two components, the sliding-window size is set as that of the maximum detection window (given in Section 4.3).

The sliding window moves forward as time passes. At each control step t, the protocol buffers, holds, and releases certain data points. As shown in Fig. 5, the workflow is as follows.

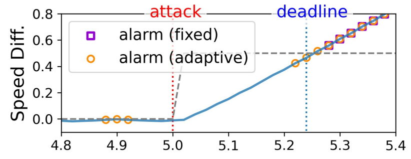
Buffer. Using the current state estimate $\bar{\boldsymbol{x}}_t$, we first calculate the residual $\boldsymbol{z}_t = |\tilde{\boldsymbol{x}}_t - \bar{\boldsymbol{x}}_t|$, where $\tilde{\boldsymbol{x}}_t = A\bar{\boldsymbol{x}}_{t-1} + B\boldsymbol{u}_{t-1}$. Then, $\bar{\boldsymbol{x}}_t$ and $\boldsymbol{z}_t$ are buffered, as shown by the blue dots. These data are within the current detection window w_c, and whether they are intact is still unknown as the detector is still checking them.

Hold. The data that has moved outside the current window is regarded trustworthy and thus held, as shown by the green dots.

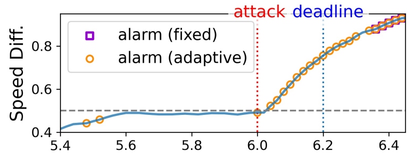
Release. The historical data before $t - w_m - 1$ are outside the sliding window and no longer need to be used, as shown by the grey dots. Thus, these data can be released to save storage space.



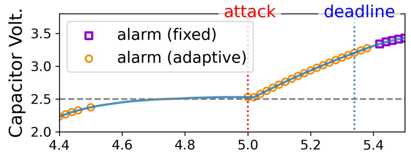
(a) Vehicle turning & bias attack



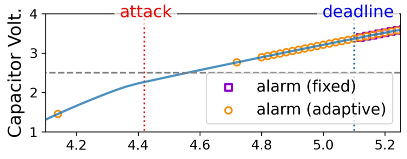
(b) Vehicle turning & delay attack



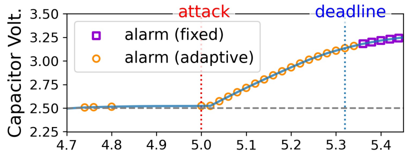
(c) Vehicle turning & replay attack



(d) RLC Circuit & bias attack



(e) RLC Circuit & delay attack



(f) RLC Circuit & replay attack

Figure 6: Comparison of detection results between adaptive window size and fixed window size for vehicle turning and RLC circuit under three attack scenarios. Blue solid line: actual system state, Grey dashed line: reference state, Red dotted vertical line: attack start time, Blue dotted vertical line: detection deadline, Orange circle marker: alert raised by adaptive detector, Purple square marker: alert raised by detector with fixed window size, Diff.:Difference, Volt.:Voltage.

Table 1: Simulation settings. No.: simulator number, δ: control step size (in seconds), PID: PID control parameters, U: control input range, ϵ: uncertainty bound, $\mathcal{S}$: safe state set, τ: detection threshold.

No.	Simulator	δ	PID	U	ϵ	$\mathcal{S}$	τ
1	Aircraft Pitch	0.02	14,0.8,5.7	$[-7, 7]$	7.8e−3	$z \in [[-\infty, -\infty, -2.5], [\infty, \infty, 2.5]]$	[0.012, 0.012, 0.012]
2	Vehicle Turning	0.02	0.5,7,0	$[-3, 3]$	7.5e−2	$z \in [-2, 2]$	[0.07]
3	Series RLC Circuit	0.02	5,5,0	$[-5, 5]$	1.7e−2	$z \in [[-3.5, -5], [3.5, 5]]$	[0.04, 0.01]
4	DC Motor Position	0.1	11,0,5	$[-20, 20]$	1.5e−1	$z \in [[-4, -\infty, -\infty], [4, \infty, \infty]]$	[0.118, 0.118, 0.118]
5	Quadrotor	0.1	0.8,0,1	$[-2, 2]$	1.56e−15	$z \in [-5, 5]$	[0.018, ..., 0.018]

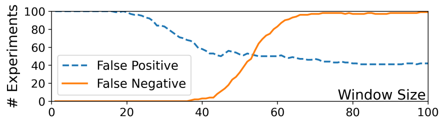


Figure 7: The number of false positive and false negative experiments changes with different window sizes.

6 EVALUATION

6.1 Simulation Setting and Results

6.1.1 Settings. We develop a simulation tool that can load different system models to simulate different physical systems. We consider 5 LTI models: aircraft pitch, vehicle turning, series RLC circuit, DC motor position, and quadrotor, which are used in [4, 8, 13, 14]. The detailed simulation setting is listed in Table 1.

We consider three sensor attack scenarios: bias, delay, and replay attack. Bias attack replaces sensor data with arbitrary values. Delay attack delays sensor measurements sent to the controller for a certain time period, so that the controller cannot update the current state estimate in time. Replay attack replaces sensor data with previously recorded ones.

6.1.2 The impact of different window size. The simulation is performed on the aircraft pitch simulator under a bias attack lasting 15 control stepsize (0.3s). We perform 100 experiments for each window size from 0 to 100. It is counted as a false positive experiment if the false positive rate exceeds 10% and counted as a false negative experiment if the attack is not detected. The number of false positive and negative experiments is plotted in Fig. 7. The result shows that the false positive number decreases and false negative number increases with increasing window size. According to Section 4.3, we choose a maximum detection window whose false negative number is acceptable for the application. Take aircraft pitch simulator as an example, we choose the maximum window size as 40, and the corresponding false negative number is only 3.

6.1.3 Results of the Adaptive Detector. We test our adaptive detection method under all 15 cases (i.e., all the combinations of 5 simulators and 3 attack scenarios). Fig. 6 shows part of the results using vehicle turning and RLC circuit simulator under bias, delay, and replay attacks. In all figures, we can see that our adaptive detector can raise alerts before the detection deadline, i.e., in-time detection, while the detector with a fixed window size finds attacks after the deadline, i.e., untimely detection. Note that our adaptive detector may raise some false alarms before real attacks are launched. This is because our adaptive detector chooses a smaller window size to catch up with the detection deadline while increasing the false positives. Note that this situation only occurs when the states are closer to the unsafe set. In practice, we consider noise in our experiments, which is also one of the reasons for false positives.

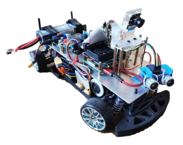
Table 2 shows all false positive and deadline miss numbers out of 100 simulations for each case. The results indicate that our adaptive detector tends to have larger false positive numbers, but with minimal deadline misses. This is because our detector will choose a smaller detection window to catch the detection deadline when the state is close to unsafe states. Note that our adaptive detector may miss the detection deadline in some cases, for instance, just 3 out of 100 experiments for DC motor under delay attack, because those attacks have a negligible effect on the physical system.

6.2 Testbed Configuration and Results

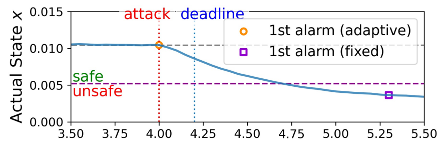
6.2.1 Testbed Configuration. We build a testbed (Fig. 8a) on a scaled RC car runing a cruise control task with a PID controller. The controller reads a magnetic rotation sensor AS5048A and computes the speed data at 20Hz. We perform system identification and get system model as $\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t, \boldsymbol{y}_t = C\boldsymbol{x}_t$, where

Table 2: Comparison of detection false positives and deadline misses with adaptive window size vs. a fixed window size. #FP: number of simulations whose false positive rate exceeds a threshold, #DM: number of simulations who miss deadline.

Simulator	Attack	Strategy	#FP	#DM
Aircraft Pitch	Bias	Adaptive	73	0
		Fixed	42	96
	Delay	Adaptive	3	0
		Fixed	2	97
	Replay	Adaptive	34	0
		Fixed	8	100
Vehicle Turning	Bias	Adaptive	71	0
		Fixed	5	34
	Delay	Adaptive	13	76
		Fixed	0	100
	Replay	Adaptive	40	10
		Fixed	3	36
Series RLC Circuit	Bias	Adaptive	54	0
		Fixed	44	65
	Delay	Adaptive	4	0
		Fixed	3	73
	Replay	Adaptive	13	0
		Fixed	5	92
DC Motor Position	Bias	Adaptive	82	0
		Fixed	47	59
	Delay	Adaptive	3	3
		Fixed	3	89
	Replay	Adaptive	3	0
		Fixed	3	46
Quadrotor	Bias	Adaptive	73	7
		Fixed	55	99
	Delay	Adaptive	6	0
		Fixed	2	87
	Replay	Adaptive	6	0
		Fixed	2	59



(a) Vehicle Testbed



(b) Attack Detection

Figure 8: Attack detection in vehicle testbed.

$A = 8.435\text{e}{-1}, B = 7.7919\text{e}{-4}, C = 3.843402\text{e}2$. The vehicle runs in a straight line at a speed of $4m/s$, and at the end of the 79^{th} step, a bias of $+2.5m/s$ is added to the speed. The safe speed range is $[2, 10]$, so safe state range is $[2/C, 10/C]$, i.e. $[5.2\text{e}{-3}, 2.6\text{e}{-2}]$. The threshold τ is set to $3.67\text{e}{-3}$. The control input range is $u \in [0, 7.7]$.

6.2.2 Testbed Results. The testbest result is shown in Fig. 8b, where the y-axis is actual states $\boldsymbol{x}$ (equals to $\boldsymbol{y}/C$), and the x-axis is time. The purple horizontal line marks the boundary between safe and unsafe states, and the detector should raise an alert before the states reach the unsafe region. The orange circle is the first alert raised by our proposed adaptive window-based detection, while the purple square is the first one raised by a fixed window-based detection (size=30). We can find our detector alert in the first step after the attack. However, the fixed window-based detection alerts after the vehicle reaching the unsafe state, which may already cause damages. Note that our adaptive detector detects the alert in the first step because the estimator computes the tightest deadline and shrinks the window size, making the average residual within the window larger than the threshold.

7 CONCLUSION

In this paper, we propose a real-time adaptive sensor attack detection system that has three key components. For Adaptive Detector, we develop a window-based detection algorithm that can dynamically adapt the detection delay and thus false alarms to meet the detection deadline and improve usability. For Detection Deadline Estimator, we develop a reachability analysis based technique to conservatively estimate the detection deadline at run time by computing the reachable set of future potential behaviors of systems. For Data Logger, we adapt a sliding-window-based data logging protocol to keep trustworthy data for deadline estimation and also sufficient data points for attack detection. Finally, we implement our detection system in multiple CPS simulators and a reduced-scale autonomous testbed to validate its efficiency and efficacy.

Acknowledgement

This research was supported in part by NSF CCF-2028740.

REFERENCES

[1] Hongjun Choi, Wen-Chuan Lee, Yousra Aafer, Fan Fei, Zhan Tu, Xiangyu Zhang, Dongyan Xu, and Xinyan Deng. 2018. Detecting attacks against robotic vehicles: A control invariant approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 801–816.

[2] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. 2018. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[3] Tianjia He, Lin Zhang, Fanxin Kong, and Asif Salekin. 2020. Exploring Inherent Sensor Redundancy for Automotive Anomaly Detection. In *57th Design Automation Conference*. ACM.

[4] Fanxin Kong, Oleg Sokolsky, James Weimer, and Insup Lee. 2019. State Consistencies for Cyber-Physical System Recovery. In *Workshop on Cyber-Physical Systems Security and Resilience (CPS-SR)*.

[5] Colas Le Guernic. 2009. *Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics*. Theses. Université Joseph-Fourier - Grenoble I. https://tel.archives-ouvertes.fr/tel-00422569

[6] Raul Quinonez, Jairo Giraldo, Luis Salazar, Erick Bauman, Alvaro Cardenas, and Zhiqiang Lin. 2020. SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants. In *29th USENIX Security Symposium (USENIX Security 20)*.

[7] Aviva Hope Rutkin. 2013. spoofers use fake GPS signals to knock a yacht off course. *MIT Technology Review* (2013).

[8] Francesco Sabatino. 2015. *Quadrotor control: modeling, nonlinearcontrol design, and simulation*. Master's thesis. KTH, Automatic Control.

[9] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. 2015. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security Symposium (USENIX Security 15)*. 881–896.

[10] David I Urbina, Jairo A Giraldo, Alvaro A Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1092–1105.

[11] Ruixuan Wang, Fanxin Kong, Hasshi Sudler, and Xun Jiao. 2021. HDAD: Hyperdimensional Computing-based Anomaly Detection for Automotive Sensor Attacks. In *27th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Brief Industry Paper Track*. IEEE.

[12] Chen Yan, Hocheol Shin, Connor Bolton, Wenyuan Xu, Yongdae Kim, and Kevin Fu. 2020. SoK: A Minimalist Approach to Formalizing Analog Sensor Security. In *2020 IEEE Symposium on Security and Privacy (SP)*. 480–495.

[13] Lin Zhang, Xin Chen, Fanxin Kong, and Alvaro A. Cardenas. 2020. Real-Time Recovery for Cyber-Physical Systems using Linear Approximations. In *41st IEEE Real-Time Systems Symposium (RTSS)*. IEEE.

[14] Lin Zhang, Pengyuan Lu, Fanxin Kong, Xin Chen, Oleg Sokolsky, and Insup Lee. 2021. Real-Time Attack-Recovery for Cyber-Physical Systems using Linear-Quadratic Regulator. In *21st ACM SIGBED International Conference on Embedded Software (EMSOFT)*.

[15] Youqian Zhang and KB Rasmussen. 2020. Detection of electromagnetic interference attacks on sensor systems. In *IEEE Symposium on Security and Privacy (S&P)*.