Variable Window and Deadline-Aware Sensor Attack Detector for Automotive CPS

Francis Akowuah
Electrical Engineering & Computer Science
South Dakota Mines
Rapid City, SD, USA
francis.akowuah@sdsmt.edu

Kenneth Fletcher
Computer Science
University of Massachusetts Boston
Boston, MA, USA
kenneth.fletcher@umb.edu

Fanxin Kong
Electrical Engineering & Computer Science
Syracuse University
Syracuse, NY, USA
fkong03@syr.edu

***Abstract*—Cyber-physical systems (CPS) are susceptible to physical attacks, and researchers are exploring ways to detect them. One method involves monitoring the system for a set duration, known as the time-window, and identifying residual errors that exceed a predetermined threshold. However, this approach means that any sensor attack alert can only be triggered after the time-window has elapsed. The length of the time-window affects the detection delay and the likelihood of false alarms, with a shorter time-window leading to quicker detection but a higher false positive rate, and a longer time-window resulting in slower detection but a lower false positive rate.**

While researchers aim to choose a fixed time-window that balances a low false positive rate and short detection delay, this goal is difficult to attain due to a trade-off between the two. An alternative solution proposed in this paper is to have a variable time-window that can adapt based on the current state of the CPS. For instance, if the CPS is heading towards an unsafe state, it is more crucial to reduce the detection delay (by decreasing the time-window) rather than reducing the false alarm rate, and vice versa. The paper presents a sensor attack detection framework that dynamically adjusts the time-window, enabling attack alerts to be triggered before the system enters dangerous regions, ensuring timely detection. This framework consists of three components: attack detector, state predictor, and window adaptor. We have evaluated our work using real-world data, and the results demonstrate that our solution improves the usability and timeliness of time-window-based attack detectors.

***Index Terms*—attack detection, sensor attack, CPS Security, Attack and Detection**

I. Introduction

Cyber-physical systems (CPS) refer to systems that are engineered through the integration of physical and computational components. These systems have found application in various fields, including healthcare, manufacturing, agriculture, aeronautics, energy, building design, civil infrastructure, transportation, and environmental quality. CPS includes systems such as smart grids, medical monitoring devices, industrial control systems, building controls, and autonomous vehicles. These systems depend on a close integration of software and hardware components. The software, also known as "cyber" components, comprises the computing and communication aspects of the system. The hardware, or "physical" components, consist of sensors that measure the system state, and actuators that produce rotary or linear motion, such as opening valves or increasing throttle.

979-8-3503-3902-4/23/$31.00 ©2023 IEEE

The integration of software and hardware in CPS has increased the vulnerability of systems that previously had closed architectures. Cyber components of CPS face attacks similar to those experienced by traditional computer software and networks, such as network eavesdropping (sniffing), buffer overflow, packet spoofing, and data modification attacks. Moreover, the physical components of CPS can be compromised, leading to the injection of malicious signals that impede the function and behavior of the system. These attacks on the physical components of CPS are referred to as *physical attacks*. Examples of physical attacks in the real world include bombarding cameras with light, injecting false GPS signals, and injecting false radar signals.

Defensive measures for CPS cyber attacks are more advanced than those for physical attacks because some conventional cyber-security solutions can also help defend CPSs against cyber attacks. Firmware hardening [1], control-flow integrity [2], and memory isolation [3] are examples of such solutions. However, they are less effective against physical attacks since these attacks do not target software/network components directly. Therefore, there is an urgent need for physical attack defense solutions.

Numerous research efforts have been undertaken to protect CPS from physical attacks, with a particular emphasis on detecting sensor attacks. However, many of the proposed solutions are impractical because they fail to adequately address the need for timely detection of attacks. One commonly used detection method involves monitoring the system for a fixed duration, known as the time-window, and identifying residual errors that exceed a predetermined threshold. However, this approach means that any alert for a sensor attack can only be triggered after the time-window has passed. The length of the time-window affects the detection delay (i.e., duration between attack launch and when it is detected) and the probability of false alarms. A shorter time-window results in faster detection but a higher false positive rate, while a longer time-window leads to slower detection but a lower false positive rate. Researchers aim to find a fixed time-window that strikes a balance between a low false positive rate and a short detection delay, but this is challenging due to a trade-off between the

two factors. In addition, selecting a fixed time-window renders it impractical for the detector to meet the detection deadlines, which refer to the time frame within which the attack must be identified before the system enters an unsafe operating state. It is crucial and necessary to ensure that the CPS does not enter unsafe operating states in the event of an attack. Detecting an attack only after destructive effects have transpired is as harmful as not having an attack detector at all.

This paper suggests an alternative solution to address the issues with a fixed time-window attack detector by proposing a variable time-window attack detector that can adjust according to the current state of the CPS. For example, if the CPS is approaching an unsafe state, reducing the detection delay (by shortening the time-window) becomes more critical than reducing the false alarm rate, and vice versa. The paper aims to demonstrate that a variable time-window detector is more practical for real-time systems and proposes a variable time-windowed framework to detect sensor attacks before the system enters an unsafe state. The framework comprises three key components: the attack detector, the state predictor, and the window adaptor. The attack detector is at the heart of the framework, using a stateful detection strategy that performs statistical tests on residuals with variable window sizes depending on the detection deadline. The state predictor uses a data model to predict sensor measurements that represent the typical behavior of the automotive CPS. Finally, the window adaptor determines the detection deadline, after which the system may enter an unsafe operating state.

The contributions of this paper are as follows: Firstly, it presents an argument supported by evidence to demonstrate that a variable-time window detector is more practical for real-time systems. Secondly, it proposes, designs, and creates a prototype for a variable-time attack detector. Thirdly, it evaluates the proposed framework's effectiveness using real world data.

The paper is structured as follows: Section II covers some preliminaries, including detection strategies and the threat model. In Section III, we describe the proposed framework's components, including the state predictor, attack detector, and window adaptor. Section IV presents our experimental setup, the conducted experiments, and the evaluation results. We delve into related work, discussion, and conclusion in Sections V, VI, and VII, respectively.

II. Preliminaries

A. Detection Strategy

As noted above, attack detectors perform statistical tests on the residuals, r_k. They raise alerts whenever the expected and observed signals significantly differ, i.e. the residual is large. Two main strategies are used for attack detection namely *stateless* and *stateful* tests [4]. In a stateless test, only one time shot is considered and the detector raises an alert for every deviation at time k (i.e. $|y_k - \hat{y}_k| = r_k \geq \tau$, where τ is a predetermined threshold). This strategy tends to produce many false alarms as the cause of the deviation may be a transient fault at that point in time. Typical examples of this are found in chi-squared ($\tilde{\chi}^2$) detectors.

Stateful strategy, on the other hand, considers multiple time-steps to determine if an alert should be raised. It maintains a statistic, S_k, that keeps track of the historical changes of r_k. An alert is raised whenever a persistent deviation is observed over multiple time-steps (i.e. $S_k \geq \tau$). Keeping track of the historical changes of r_k can be done in multiple ways such as (1) using change detectors (2) taking an exponential weighted moving average (EWMA) and (3) taking an average over a time-window. We focus on detectors that use the stateful strategy and use a fixed-time window.

B. Threat Model

In our threat model, we consider the possibility of an adversary compromising the sensor through physical attacks that introduce interfering signals, such as magnetic fields or light, into the sensor's physical environment. Such an attack would compromise the accuracy and integrity of the sensor's measurements, leading to the transmission of incorrect system state information to the controller. This false sensor data would have a ripple effect on both the control input computed by the controller and the control output performed by the actuator, ultimately causing the system to deviate from its reference state.

Furthermore, we make the assumption that the attacker has no access to the control program and proposed framework components running on-board. As mentioned earlier, we do not center our attention on cyber attacks that target the software and communication components, as these can be efficiently protected using existing software security techniques like CFI. Instead, our focus is entirely on sensor attacks. It should be noted that this paper does not address attacks aimed at non-vehicle control logic, such as the computer vision system of the automotive CPS.

C. Framework Overview

The proposed framework's components work together to signal an alert for a sensor attack before the CPS enters an unsafe operating state. Figure 1 presents the end-to-end framework of our suggested variable-window real-time sensor attack detector. The state predictor component anticipates the expected sensor measurements during system operation. The window adaptor component calculates the detection deadline and window length (detection delay) that enable the framework to meet detection deadlines. The attack detector component receives input from the window adaptor and the state predictor and executes the following tasks: (1) calculates the residual r_k (i.e., the difference between the expected and observed values). If r_k is significant, it obtains the window size from the window adaptor as input, (2) the window size input is employed in the detection strategy to decide whether to raise an alarm. It should be noted that the selection of variable window sizes allows our detector framework to adapt its behavior to meet detection deadlines.

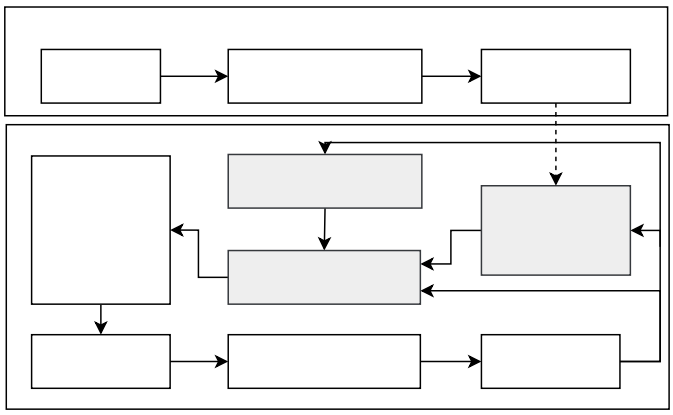


Fig. 1: System design of variable window real-time sensor attack detection framework.

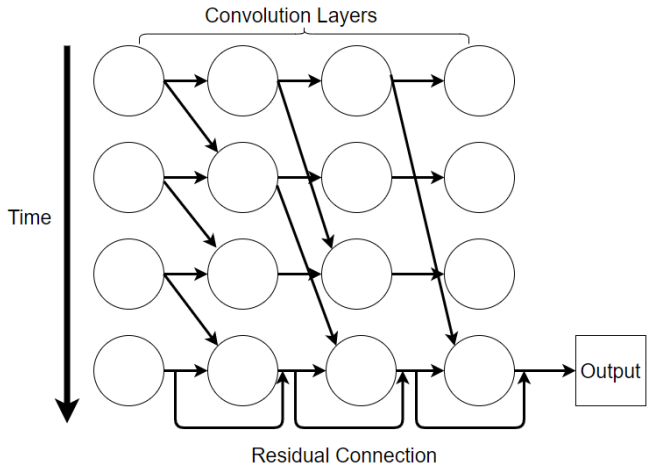


Fig. 2: Temporal Convolutional Networks structure

III. Design

A. Design of Attack Detector

We present the design of the attack detector component in this section. The component is responsible for performing the attack detection strategy of our framework utilizing inputs from the Window Adaptor (§III-C) and the State Predictor (§III-B) components.

We formulate the attack detection problem as follows. Given the predicted (expected) sensor value $\hat{y}_t \in \mathbb{R}^n$, the observed sensor reading $y_t \in \mathbb{R}^n$, a predetermined threshold τ, and the window length l, we want to determine the appropriate time to raise an alarm t_{alarm} before the system touch unsafe state:

$$t_{alarm} = \sum_{t=k-l+1}^{k} r_t > \tau \tag{1}$$

where $r_t = |y_t - \hat{y}_t|$. Note here that the moving sum of the residuals is taken over a window $[k-l+1, k]$. The $\hat{y}_t$ used in Eqn. 1 is the output of the state predictor which has the responsibility of predicting expected sensor values. The window adaptor provides the window length l to be used in the detection strategy.

B. Design of State Predictor

In this subsection, we describe the design of the state predictor component. This component is responsib for forecasting system states by using historical data from a fixed-size sliding window, i.e., fixed-size reception fields. We utilize the Temporal Convolutional Network (TCN) [5] for this purpose.

TCN is a convolutional sequence prediction architecture specifically designed to capture action segmentation in time-series initially [6], [7]. Recently, several studies have demonstrated that TCN outperforms RNN-based structures on various time-series tasks [8]–[11]. TCNs have the ability to take a sequence of any length and map it to an output sequence of equal length, just like an RNN. TCNs differ from RNNs in that the convolutions in TCNs are causal (i.e., there is no information "leakage" from the future to the past). Additionally, TCNs can be trained easily in parallel since the network does not contain gate components.

We opted to use TCN instead of memory-based RNNs like LSTM and GRU as our state predictor component for the following reasons:

- Firstly, TCN delivers higher prediction accuracy than LSTM, meaning that the generated predictions are closer to the actual system state.
- Secondly, research has demonstrated that TCN has a shorter inference time than LSTM [12], which can enhance the framework's decision-making speed by enabling faster inference of sensor speeds. Consequently, using TCN can achieve a shorter detection delay.
- Lastly, because TCN can be considerably deep, it can memorize longer histories than LSTM. A longer memory is beneficial in generating better predictions when there are long-term dependencies between data and the time horizon.

Fig. 2 illustrates the architecture of TCN. With each layer, the prediction for each step benefits from a range of time dependencies of varying lengths. As a result, TCN has a relatively long memory that increases with depth. Moreover, TCN does not include any intricate components like LSTM gates, which suggests that TCN should have a faster inference speed than LSTM while maintaining the same level of accuracy.

Given a previously observed input sequence $x_0, x_1, ..., x_T$, we can use TCN to predict a corresponding output sequence $y_0, y_1, ..., y_T$. This kind of sequence modeling network is any function $f : \mathcal{X}^{T+1} \rightarrow \mathcal{Y}^{T+1}$ that produces a mapping

$$\hat{y_0}, \hat{y_1}, ..., \hat{y_T} = f(x_0, x_1, ..., x_T) \tag{2}$$

if it satisfies the causal constraint that y_t depends only on $x_0, x_1, ..., x_t$ and not on any future inputs $x_{t+1}, x_{t+2}, ..., x_T$ [5].

The TCN relies on two key principles: the output should have the same length as the input, and information from future time steps should not impact past predictions. To fulfill these

principles, TCN employs a 1D fully-convolutional network (FCD). However, a basic convolution can only examine historical data within a window that is proportional to the network's size. Therefore, TCN utilizes dilated convolutions [5]. The dilated convolution operation F is applied to each element s of a 1-D sequence input x$\in \mathbb{R}^n$ and a filter $f : \{0, 1, ..., k-1\} \rightarrow \mathbb{R}$. It can be defined as:

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i} \tag{3}$$

where d is the dilation factor, k is the filter size, and $s - d \cdot i$ accounts for the direction of the past.

Fig. 2 demonstrates that each layer in the TCN has a distinct dilation rate s, which represents the distance between convolutional operations and enables the network to select which time steps to apply convolution to. In addition, a residual connection is used to combine the input to the layer and the convolutional signal. The dilated output $\hat{S}_t^{(l)}$ of the l^{th} layer at time step t, and the output $S_t^{(l)}$ after applying residual connections, can be expressed as [7]:

$$\hat{S}_t^{(l)} = f\left(W^{(1)} S_{t-s}^{(l-1)} + W^{(2)} S_t^{(l-1)} + b\right) \tag{4}$$

$$S_t^{(l)} = S_t^{(l-1)} + V \hat{S}_t^{(l)} + e \tag{5}$$

where $W^{(i)}$ denotes the i^{th} convolution filter in the layer, b denotes the bias vector, v and e denotes the weights and bias vector of the residual.

Pre-processing and Sensor correlation

In cyber-physical systems, it is common to use multiple types of sensors for tasks such as function execution and environmental navigation. For instance, modern vehicles are equipped with various sensors including wheel speed, engine speed, GPS, and boost pressure. While these sensors may monitor different physical phenomena, their measurements can exhibit correlation [13]. For example, when the accelerator is pressed, it is likely that both the wheel speed and engine speed will increase. Thus, we can say that the accelerator, wheel speed, and engine speed sensors are correlated. Our objective is to leverage or capture these correlations in our model. Therefore, we aim to select the most correlated sensors to train our model. While we may know of some correlated sensors from our experience with driving, we seek to confirm these correlations using the Pearson's Correlation Coefficient (PCC) algorithm, as well as discover new correlations that are not obvious to our domain knowledge. The result of the PCC algorithm is a correlation matrix depicted in Figure 3.

PCC is a linear correlation measure between two sets of variables, with values ranging from +1 to -1. If X and Y are two variables, a PCC value of +1 indicates that when X increases, Y also increases, while a PCC value of -1 indicates that when X decreases, Y also decreases. A PCC value of 0 means that there is no linear relationship between X and Y. As illustrated in Fig. 3, the vehicle speed, engine speed, and wheel speed sensors are strongly correlated, while the

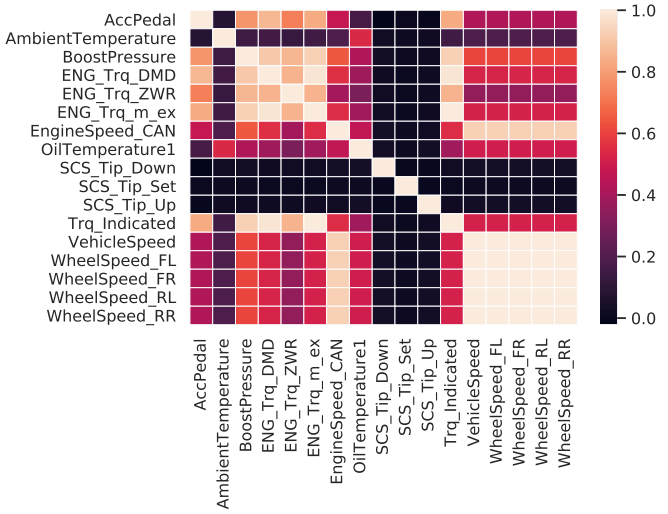


Fig. 3: Example confirming the wheel speed sensors in the dataset has strong correlation with the wheel speed, engine speed and boost pressure sensors.Table I shows the available sensors in the dataset.

SCS_Tip_Up sensor has no correlation with the wheel speed sensor. We chose sensors with PCC values above +0.6 to train our model.

C. Design of Window Adaptor

This component serves the purpose of determining the optimal length of the time window to be utilized in the attack detector. It is comprised of two subsystems, namely the deadline estimator and the window-length analyzer, which are elaborated upon in the following sections.

1) Deadline Estimator: As mentioned earlier, the effectiveness of attack detectors is not solely dependent on correctly raising attack alarms, but also on detecting them in a timely manner, that is, before the system enters unsafe states. Detecting attacks after the system has entered unsafe states is deemed potentially worthless and can result in harmful consequences.

In this paper, we refer to the estimated time in the future when the system might reach the unsafe set as the *detection deadline*. It is important to note that our proposed framework does not depend on any particular method for estimating the deadline.

The choice of a deadline estimation method typically relies on whether system dynamics information or system data is available. In [14], the authors utilized system dynamics information to perform reachability analysis, which was then used to compute the deadline. Although this approach can provide more accurate reachable set approximations, it is not applicable in many important applications where detailed system dynamics information is either unavailable or only accessible through simulations. Consequently, data-driven approaches have gained attention from researchers. For example, in [15], the authors proposed a data-driven deadline estimation method that utilizes only system data to calculate the maximum change

rate of the sensor value and estimate the shortest time when the system might reach the unsafe set. Recently, many data-driven reachability analysis methods such as [16]–[22] have been proposed; however, most of these methods are computationally expensive and only applicable in offline scenarios, making them unsuitable for real-time systems like cyber-physical systems. For example, in [16], an experiment that computed the reachable set approximation took 39 minutes. To address this issue, we propose a purely data-driven deadline estimation method that is not computationally expensive and therefore well-suited for resource-constrained systems.

The proposed method involves using data-driven reachability analysis to obtain approximations of the reachable set and subsequently checking if the unsafe set is encompassed within this set. If this condition is met, the minimum time required for the system to reach an unsafe state is identified as the detection deadline.

We define the data-driven reachability analysis as a time series forecasting problem where we predict sensor values over a finite time horizon $[t_0, t_1]$. To this end, we adopt Autoregressive Integrated Moving Average (ARIMA) as the technique for predicting sensor values. ARIMA generalizes Autoregressive Moving Average (ARMA) which combines the Autoregressive (AR) process and Moving Average (MA) processes to build a composite model of the time series.

An AR model of order p, i.e., $AR(p)$, can be represented as a linear process by:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t \tag{6}$$

where the terms in ϕ_i are autocorrelation coefficients at lags 1,2...p, c is a constant and ϵ_t is white noise.

The MA model of order q, i.e., $MA(q)$ is represented as

$$y_t = \mu + \sum_{i=0}^{q} \theta_i \epsilon_{t-i} \tag{7}$$

where μ refers to the expectation of y_t, θ is weight that is applied to the current and previous values of a stochastic term in the time series. ϵ_t is the Gaussian white noise. Thus, an ARIMA model of order (p, q, d) is formed by adding the AR and MA models as:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t + \sum_{i=0}^{q} \theta_i \epsilon_{t-i} \tag{8}$$

The parameters of the ARIMA model $ARIMA(p, q, d)$ are defined as follows: p : refers to the number of lag observations included in the model (it is also known as the lag order). d represents the number of times that the raw observations are differenced and q is the size of the moving average window, also called the order of moving average.

Once the ARIMA model is trained from offline data, it can be used online to predict sensor values or the reachable set $\hat{R}_{[t_0,t_{reach}]}$ which is defined as:

$$\hat{R}_{[t_0,t_{reach}]} = \hat{y}(t_0), \hat{y}(t_1), \hat{y}(t_2) \dots \hat{y}(t_{reach}) \tag{9}$$

Let Λ represent the unsafe set associated with sensor. Hence the system is considered safe over the finite time horizon, $[t_0, t_1]$, if $\hat{R}_{[t_0,t_{reach}]} \cap \Lambda = \emptyset$. Otherwise, it can be asserted that the reachable set contains unsafe set. Therefore, we proceed to the next step to find the minimum time t_d ($t_0 \geq t_d \leq t_{reach}$) that is associated with an unsafe state within the reachable set $\hat{R}_{[t_0,t_{reach}]}$. t_d thus, become the detection deadline.

2) Window-length Analyzer: This component determines the appropriate window length to be used in the detection strategy. Remember that the choice of time-window length dictates a trade-off between false alarm rate and detection delay. This component enables the framework to bias detection delay and false alarm rate. One of the goals of the proposed attack detector framework is to meet the attack detection deadline.

This component functions in two phases. The offline phase profiles the CPS to build a lookup table that establishes the relationship between the time-window length, and the detection delay. This phase is performed only once for the CPS. During the online phase, to perform its online adaptive functionality, the window analyzer queries the lookup table to output the time window length that adjusts the detection delay to meet the given detection deadline.

IV. Evaluation

In this section, we evaluate the effectiveness of the proposed solution. Firstly, we assess the state predictor's capability to learn the nominal system behavior. Secondly, we compare our variable approach with the fixed window approach. Lastly, we measure the false-positive and false-negative rates.

A. Experimental Setup

Our implementation of the deep learning model was carried out in Python, utilizing the PyTorch Deep Learning framework. To train the model, the dataset was split into three parts: training set, validation set, and test set, with proportions of 60%, 20%, and 20%, respectively. We obtained our dataset from the AEGIS Big Data Project [23][1], a publicly-available real-world automotive CAN bus dataset, which includes sensor data sampled at 20Hz. The types of sensors used are listed in Table I.

B. Experiments and results

Experiment I: This experiment aims to evaluate the state prediction capability of the state predictor and the degree to which the model has captured the nominal system behavior. The accuracy of state prediction indicates the effectiveness of the model in capturing the system's behavior. The prediction results for the wheel speed and engine revolution sensor values are presented in Fig.4 and Fig.5, respectively. The figure shows a close match between the actual measurements and the forecast lines, indicating that the model has effectively captured the nominal system behavior. As shown in Fig. 6, the model incurred near-zero errors on average, approximating the

[1]https://zenodo.org/record/3267184#.X5YtpIhKg2x

TABLE I: Some sensors in the dataset used in experiment.

CAN bus Sensors	GPS Sensors	IMU Sensors
ASR	Acceleration	Accelerometer_X
AccPedal	Current_sec	Accelerometer_Y
AirIntakeTemperature	Direction	Accelerometer_Z
AmbientTemperature	Distance	Body_acceleration_X
BoostPressure	Velocity	Body_acceleration_Y
BrkVoltage		Body_acceleration_Z
EngineSpeed_CAN		G_force
EngineTemperature		Magnetometer_X
Kickdown		Magnetometer_Y
MFS_Tip_Down		Magnetometer_Z
MFS_Tip_Up		Velocity_X
SteerAngle		Velocity_Y
Trq_FrictionLoss		Velocity_Z
Trq_Indicated		
VehicleSpeed		
WheelSpeed_FL		
WheelSpeed_FR		
WheelSpeed_RL		
WheelSpeed_RR		
Yawrate		

ASR = Acceleration Slip Regulation, ACC = Acceleration, BRK = Break, MFS = Misfiring System, TRQ = Torque, FL = Front Left, FR = Front Right, RL = Rear Left, RR = Rear Right, G = Gravity

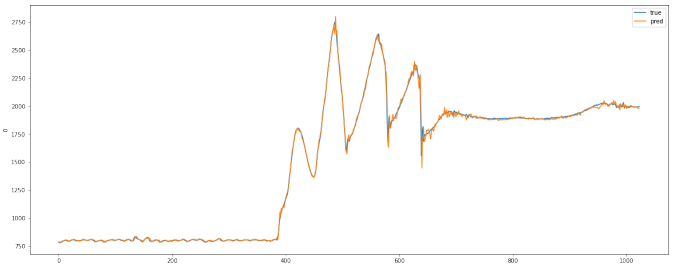


Fig. 5: The state predictor forecasting the engine speed sensor.

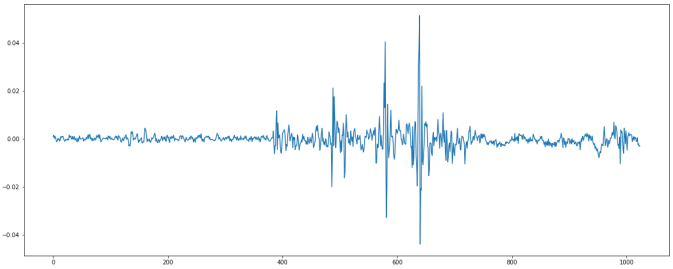


Fig. 6: The error between the state predictor's forecasting of the engine speed sensor.

system behavior. However, to further improve the model, it is recommended to use additional training data containing more nominal system data

Experiment II: In this experiment, we measure the false-positive (FP) and false-negative (FN) rates of the detector. First, we embed ten simulated attack ranges in the test data set. The first attack range compromises the observed sensor measurement by adding 0.2 km/h. The magnitude of subsequent attack ranges increases by 0.2 km/h i.e. the second attack range adds 0.4 km/h to the data, the third 0.6 km/h and so on.

Second, the detector is tasked to detect the ten attacks under different monitoring parameters (window length and threshold). Fig. 7 shows the results of FP. It can be observed that as the window length gets larger, the FP decreases. This observation is due to normalization of the accumulated errors before the threshold comparison. Hence, a larger window results in smaller normalized errors. On the other hand, the FN is observed in Fig. 8 to increase as the window get larger. The same reason for the FP results also attributes to the FN observation. These two observations show the detector can vary its behavior by varying the window length and still achieve acceptable low FP and FN.

Experiment III: This experiment shows how the proposed framework allows for flexible detection delays to be achieved by the detector, thereby enabling it to meet detection deadlines. To compare the effectiveness of our variable-window detector with a fixed-window attack detector solution, such as the one described in [24], we consider an attack scenario where the attack is initiated at 10s and the detection deadline is set at 16s, beyond which the system enters an unsafe region. The results

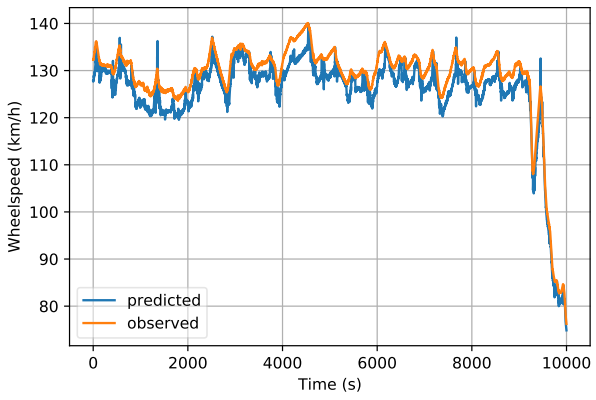


Fig. 4: The state predictor forecasting the wheel speed sensor.

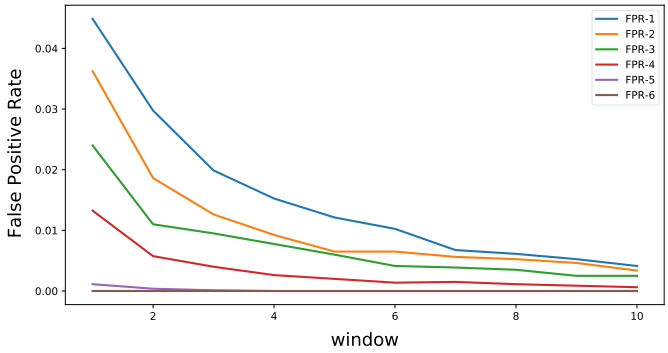


Fig. 7: The false-positive rate measurements under various monitoring parameters. FPR-1 means a threshold of 1. FPR-2 means a threshold of 2 and so on.

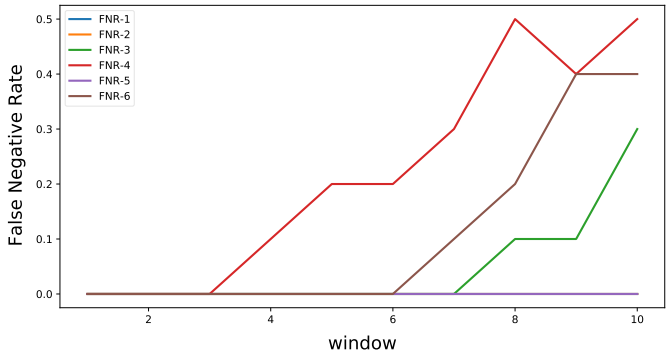


Fig. 8: The false-negative rate measurements under various monitoring parameters. FNR-1 means a threshold of 1. FNR-2 means a threshold of 2 and so on.

of the experiment and the attack scenario are illustrated in Fig. 10.

The figure shows that the length of the time window influences the time taken to detect the attack, or in other words, the detection delay. Each window length corresponds to a specific detection delay; for example, Fig.10a resulted in a detection delay of 11s, whereas Fig.10e resulted in a detection delay of 15s. Additionally, we observe from the figure that the fixed-time-window detector consistently has the same detection delay (17.5s in our experiment), making it incapable of meeting varying detection deadlines. In contrast, our variable-window detector framework can adapt to different detection deadlines by selecting appropriate window lengths.

V. Related Work

Given the indispensable roles that cyber-physical systems play in modern society, their security issues have received due attention, especially from the research community. Broadly, researchers have addressed (1) CPS vulnerabilities and exploitation [25]–[35], (2) preventing CPS attacks [29], [30], [36]–[39], (3) detecting CPS attacks [4], [40]–[46], and mitigating CPS attacks [14], [34], [47]–[51]. The general mechanism for attack detection solutions include observing the internal state of a CPS subsystem or the CPS as a whole. Other solutions monitor the interaction of the CPS components in order to spot any malicious or anomalous activities. Lastly, the CPS can be monitored through out-of-band channels such as radio frequency emissions from a CPS component [42]. Commonly, researchers create models from system data or system dynamics information that captures the normal behavior of the system. The models are used to predict the expected behavior of the CPS. The predictions are compared with observed system state data to determine any on-going malicious activity. Fig. 9 shows this general strategy for attack detection.

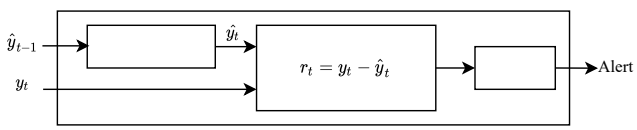


Fig. 9: The general attack detection strategy

Existing works have captured the dynamics of the systems in various ways. Some treat the system as a black box by building a data model from system data such as sensor measurement, system logs, and command inputs and outputs. Such works include but not limited to [13], [15], [52]–[55]. The state predictor in our framework follows this approach to model the physics of the system.

In other research works, the approximate knowledge about the physical dynamics (physical invariant) of the system are known in terms of the set of equations. Therefore, they treat the system as a greybox by learning the parameters of the set of equations utilizing techniques such as system identification (SI). Works in this direction have considered the system as a linear dynamical system (LDS) [24] or as a non-linear system [56].

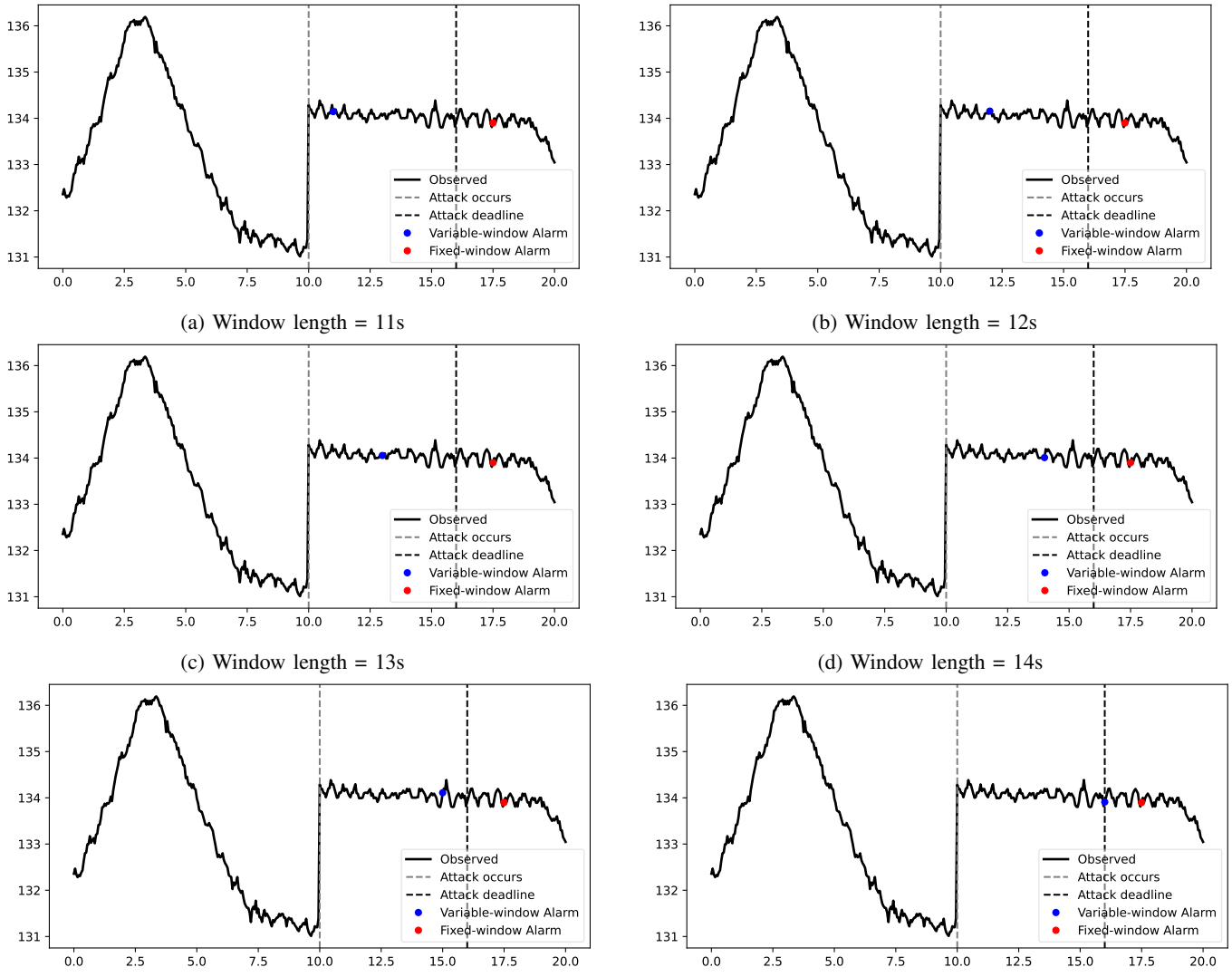
The sensor measurements in a system have correlations among them as a result of physical laws. Whenever the correlation is violated it is an indication of an attack or fault. Existing works have employed various techniques to exploit the correlation in their attack detection mechanism. Some researchers have used homogeneous sensors [**?**] for correlation exploitation whereas others have used heterogeneous sensors [13], [52], [57], [58]. Our framework also exploits correlation among heterogeneous sensors. Unlike, existing works, we address the timing constraints of attack detection.

Although there are many attack detection works, the issue of raising an alert in a timely manner remain largely unexplored. Most of the literature have focused on correctly detecting an attack without considering the timeliness of doing so. To the best of our knowledge, [15] is one of the attack detection works that addresses the real-time needs of attack detectors. Their real-time attack detection framework selects the appropriate CUSUM bias parameter that enables the detector to raise an alert before the system touch unsafe regions. The solution provided in [15], however, is inapplicable to time-window-based sensor attack detectors since their CUSUM-based solution do not utilize time windows. Our work fills this gap by providing the real-time needs of the numerous time-window sensor attack detectors. We note that the authors in [14], [59] also consider real-time issue in autonomous CPS security, however, their work focuses on attack recovery instead of attack detection. In brevity, [14], [59] devise a reactive solution that seeks to ensure that the CPS attack is mitigated before the system plunge into unsafe operating states.

VI. Discussion

A. Attack Recovery

In ensuring the security of Cyber-Physical Systems (CPS), sensor attack detection is a vital step as it helps identify any malicious activity. The output of attack detectors is usually an alarm. However, it is equally important to consider what steps should be taken after the alarm has been raised. While humans can often mitigate the attack after detection in most practical



(a) Window length = 11s

(b) Window length = 12s

(c) Window length = 13s

(d) Window length = 14s

(e) Window length = 15s

(f) Window length = 16s

Fig. 10: Comparing our framework with a fixed time-window approach. In all scenarios, the fixed-time-window detector raises the alarm at 17.5s always. Our approach enables varying the window length such that alarms can be raised before the detection deadline.

situations, it is crucial to have automatic attack recovery measures in place to restore the system or enable it to operate safely in critical modes.

While the focus of this paper is on timely attack detection, it does not cover attack recovery. However, the proposed solution can be integrated with attack recovery solutions, such as those discussed in [14], [50], [59], to enhance the overall security of CPS.

B. Limitations

The effectiveness of the proposed solution presented in this paper relies heavily on accurately estimating the detection deadline. Most methods for estimating deadlines depend on reachability analysis, which is a popular and effective way to ensure safety in uncertain situations. Reachability analysis characterizes all possible system evolutions. However, despite significant progress in the study of reachability analysis, computing the exact reachable state is still an open problem. It is worth noting that our proposed framework does not rely on any particular deadline estimation method. However, since the best available deadline estimation method provides an approximation rather than an exact reachable set, our framework may make decisions based on erroneous detection deadlines in some cases.

Finally, the proposed framework is dependent on accurately modeling system dynamics to predict expected system behavior. However, the framework is not reliant on any specific system model. In this work, we proposed building the system model using system data, as obtaining system dynamic information can often be challenging. With sufficient training

data, the model should be able to accurately estimate the system state. However, it can be difficult to obtain training data that covers every system behavior necessary for the model to learn and capture accurately. Online learning techniques may be useful to continue to train the model as new data is generated. It is important to note, however, that in online learning, an attacker may inject malicious signals that the model could learn as normal behavior. Distinguishing between normal system behavior and malicious, stealthy signals for online model learning remains an open problem.

C. Applicability to other CPS domain

This paper presents a solution that specifically targets the autonomous cyber-physical system domain. It is worth noting that the proposed solution can also be applied to other CPS domains, such as industrial control systems and smart grids. In more stable CPS environments where the system behavior is more predictable, the estimation of the detection deadline can be more accurately computed, resulting in more reliable and timely attack detection.

VII. Conclusion

Modern society relies on cyber-physical systems for safety-critical functions. However, the integration of information technology with physical processes has exposed these once-isolated systems to a variety of attacks. Traditional cybersecurity solutions are not sufficient against physical attacks, and existing sensor attack detection solutions do not address the timing and usability of attack detectors adequately. To address these issues, this paper proposes a variable-time window detector that can adjust its metrics based on the current state of the CPS to meet the detection deadline. Unlike fixed time-window detectors, our proposed framework adapts its behavior to meet attack detection deadlines. This framework consists of three components: attack detector, state predictor, and window adaptor. We have evaluated our work using real-world data, and the results demonstrate that our solution improves the usefulness and timeliness of time-window-based attack detectors. This leads to more reliable and robust cyber-physical systems. Furthermore, we emphasize that our proposed solution is applicable to other CPS domains, such as industrial control systems and smart grids, and can be particularly effective in stable CPS where the system evolution is more predictable, enabling more accurate deadline estimation.

Acknowledgment

This work was supported in part by NSF CNS-2143256, SD Mine's Nelson Research award fund, and SDBOR CRG award fund. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the National Science Foundation (NSF) or South Dakota Board of Regents (SDBOR).

References

[1] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space randomization," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 298–307.

[2] A. A. Clements, N. S. Almakhdhub, K. S. Saab, P. Srivastava, J. Koo, S. Bagchi, and M. Payer, "Protecting bare-metal embedded systems with privilege overlays," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 289–303.

[3] C. H. Kim, T. Kim, H. Choi, Z. Gu, B. Lee, X. Zhang, and D. Xu, "Securing real-time microcontroller systems through customized memory view switching." in *NDSS*, 2018.

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

[5] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[6] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *European Conference on Computer Vision*. Springer, 2016, pp. 47–54.

[7] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[8] J. Yan, L. Mu, L. Wang, R. Ranjan, and A. Y. Zomaya, "Temporal convolutional networks for the advance prediction of enso," *Scientific reports*, vol. 10, no. 1, pp. 1–15, 2020.

[9] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[10] P. Hewage, A. Behera, M. Trovati, E. Pereira, M. Ghahremani, F. Palmieri, and Y. Liu, "Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station," *Soft Computing*, vol. 24, no. 21, pp. 16 453–16 482, 2020.

[11] B. Martinez, P. Ma, S. Petridis, and M. Pantic, "Lipreading using temporal convolutional networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6319–6323.

[12] P. Lara-Benítez, M. Carranza-García, J. M. Luna-Romera, and J. C. Riquelme, "Temporal convolutional networks applied to energy-related time series forecasting," *Applied Sciences*, vol. 10, no. 7, p. 2322, 2020.

[13] T. He, L. Zhang, F. Kong, and A. Salekin, "Exploring inherent sensor redundancy for automotive anomaly detection," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[14] L. Zhang, X. Chen, F. Kong, and A. A. Cardenas, "Real-time recovery for cyber-physical systems using linear approximations," in *41st IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2020.

[15] F. Akowuah and F. Kong, "Real-time adaptive sensor attack detection in autonomous cyber-physical systems," in *27th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2021.

[16] A. Devonport, F. Yang, L. E. Ghaoui, and M. Arcak, "Data-driven reachability analysis with christoffel functions," *arXiv preprint arXiv:2104.13902*, 2021.

[17] Y. Yang, J. Zhang, K.-Q. Cai, and M. Prandini, "Multi-aircraft conflict detection and resolution based on probabilistic reach sets," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 309–316, 2016.

[18] D. Ioli, A. Falsone, H. Marianne, B. Axel, and M. Prandini, "A smart grid energy management problem for data-driven design with probabilistic reachability guarantees," in *4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, vol. 48, 2017, pp. 2–19.

[19] H. Sartipizadeh, A. P. Vinod, B. Açikmeşe, and M. Oishi, "Voronoi partition-based scenario reduction for fast sampling-based stochastic reachability computation of linear systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 37–44.

[20] A. Devonport and M. Arcak, "Estimating reachable sets with scenario optimization," in *Learning for dynamics and control*. PMLR, 2020, pp. 75–84.

[21] A. Alanwar, A. Koch, F. Allgöwer, and K. H. Johansson, "Data-driven reachability analysis from noisy data," *arXiv preprint arXiv:2105.07229*, 2021.

[22] ——, "Data-driven reachability analysis using matrix zonotopes," in *Learning for Dynamics and Control*. PMLR, 2021, pp. 163–175.

[23] C. Kaiser, A. Stocker, and A. Festl, "Automotive CAN bus data: An Example Dataset from the AEGIS Big Data Project," Jul. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3267184

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

[25] Y.-L. Huang, A. A. Cárdenas, S. Amin, Z.-S. Lin, H.-Y. Tsai, and S. Sastry, "Understanding the physical and economic consequences of attacks on control systems," *International Journal of Critical Infrastructure Protection*, vol. 2, no. 3, pp. 73–83, 2009.

[26] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 103–117.

[27] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 881–896.

[28] J. Selvaraj, G. Y. Dayanıklı, N. P. Gaunkar, D. Ware, R. M. Gerdes, and M. Mina, "Electromagnetic induction attacks against embedded systems," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 499–510.

[29] I. Giechaskiel and K. Rasmussen, "Taxonomy and challenges of out-of-band signal injection attacks and defenses," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 645–670, 2019.

[30] K. Fu and W. Xu, "Risks of trusting the physics of sensors," *Communications of the ACM*, vol. 61, no. 2, pp. 20–23, 2018.

[31] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 129–142.

[32] S. Amin, X. Litrico, S. Sastry, and A. M. Bayen, "Cyber security of water scada systems—part i: Analysis and experimentation of stealthy deception attacks," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1963–1970, 2012.

[33] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "Revealing stealthy attacks in control systems," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2012, pp. 1806–1813.

[34] S. Amin, A. A. Cárdenas, and S. S. Sastry, "Safe and secure networked control systems under denial-of-service attacks," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2009, pp. 31–45.

[35] M. Krotofil, A. Cardenas, J. Larsen, and D. Gollmann, "Vulnerabilities of cyber-physical systems to stale data—determining the optimal time to launch attacks," *International journal of critical infrastructure protection*, vol. 7, no. 4, pp. 213–232, 2014.

[36] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, "Key management systems for sensor networks in the context of the internet of things," *Computers & Electrical Engineering*, vol. 37, no. 2, pp. 147–159, 2011.

[37] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu, "They can hear your heartbeats: Non-invasive security for implantable medical devices," in *Proceedings of the ACM SIGCOMM 2011 conference*, 2011, pp. 2–13.

[38] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu, "Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks," in *2017 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2017, pp. 3–18.

[39] J. H. Castellanos, D. Antonioli, N. O. Tippenhauer, and M. Ochoa, "Legacy-compliant data authentication for industrial control system traffic," in *International Conference on Applied Cryptography and Network Security*. Springer, 2017, pp. 665–685.

[40] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A.-R. Sadeghi, and M. Schunter, "Sana: Secure and scalable aggregate network attestation," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 731–742.

[41] K. Paridari, N. O'Mahony, A. E.-D. Mady, R. Chabukswar, M. Boubekeur, and H. Sandberg, "A framework for attack-resilient industrial control systems: Attack detection and controller reconfiguration," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 113–128, 2017.

[42] T. Shekari, C. Bayens, M. Cohen, L. Graber, and R. Beyah, "Rfdids: Radio frequency-based distributed intrusion detection system for the power grid." in *NDSS*, 2019.

[43] Q. Gu, D. Formby, S. Ji, H. Cam, and R. Beyah, "Fingerprinting for cyber-physical system security: Device physics matters too," *IEEE Security & Privacy*, vol. 16, no. 5, pp. 49–59, 2018.

[44] W. Aoudi, M. Iturbe, and M. Almgren, "Truth will out: Departure-based process-level detection of stealthy attacks on control systems," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 817–831.

[45] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, 2015, pp. 13–24.

[46] V. P. Illiano, R. V. Steiner, and E. C. Lupu, "Unity is strength! combining attestation and measurements inspection to handle malicious data injections in wsns," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2017, pp. 134–144.

[47] Y. Mo and B. Sinopoli, "Secure estimation in the presence of integrity attacks," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 1145–1151, 2014.

[48] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure estimation and control for cyber-physical systems under adversarial attacks," *IEEE Transactions on Automatic control*, vol. 59, no. 6, pp. 1454–1467, 2014.

[49] D. Wijayasekara, O. Linda, M. Manic, and C. Rieger, "Fn-dfe: Fuzzy-neural data fusion engine for enhanced resilient state-awareness of hybrid energy systems," *IEEE transactions on cybernetics*, vol. 44, no. 11, pp. 2065–2075, 2014.

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

[51] F. Akowuah, R. Prasad, C. O. Espinoza, and F. Kong, "Recovery-by-learning: Restoring autonomous cyber-physical systems from sensor attacks," in *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2021, pp. 61–66.

[52] H. Li, L. Zhao, M. Juliato, S. Ahmed, M. R. Sastry, and L. L. Yang, "Poster: Intrusion detection system for in-vehicle networks using sensor correlation and integration," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 2531–2533.

[53] F. van Wyk, Y. Wang, A. Khojandi, and N. Masoud, "Real-time sensor anomaly detection and identification in automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1264–1276, 2019.

[54] A. R. Javed, M. Usman, S. U. Rehman, M. U. Khan, and M. S. Haghighi, "Anomaly detection in automated vehicles using multistage attention-based convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[55] J. Shin, Y. Baek, J. Lee, and S. Lee, "Cyber-physical attack detection and recovery based on rnn in automotive brake systems," *Applied Sciences*, vol. 9, no. 1, p. 82, 2019.

[56] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "{SAVIOR}: Securing autonomous vehicles with robust physical invariants," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 895–912.

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

[58] F. Guo, Z. Wang, S. Du, H. Li, H. Zhu, Q. Pei, Z. Cao, and J. Zhao, "Detecting vehicle anomaly in the edge via sensor consistency and frequency characteristic," *IEEE Transactions on Vehicular Technology*, 2019.

[59] L. Zhang, P. Lu, F. Kong, X. Chen, O. Sokolsky, and I. Lee, "Real-time attack-recovery for cyber-physical systems using linear-quadratic regulator," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–24, 2021.