Cyber Security of Sensor Systems for State Sequence Estimation: an AI Approach

Xubin Fang, Rick S. Blum, *Life Fellow, IEEE* , Ramesh Bharadwaj, and Brian M. Sadler *Life Fellow, IEEE*

***Abstract*—Sensor systems are extremely popular today and vulnerable to sensor data attacks. Due to possible devastating consequences, counteracting sensor data attacks is an extremely important topic, which has not seen sufficient study. This paper develops the first methods that accurately identify/eliminate only the problematic attacked sensor data presented to a sequence estimation/regression algorithm under a powerful attack model constructed based on known/observed attacks. The approach does not assume a known form for the statistical model of the sensor data, allowing data-driven and machine learning sequence estimation/regression algorithms to be protected. A simple protection approach for attackers not endowed with knowledge of the details of our protection approach is first developed, followed by additional processing for attacks based on protection system knowledge. In the cases tested for which it was designed, experimental results show that the simple approach achieves performance indistinguishable, to two decimal places, from that for an approach which knows which sensors are attacked. For cases where the attacker has knowledge of the protection approach, experimental results indicate the additional processing can be configured so that the worst-case degradation under the additional processing and a large number of sensors attacked can be made significantly smaller than the worst-case degradation of the simple approach, and close to an approach which knows which sensors are attacked, for the same number of attacked sensors with just a slight degradation under no attacks. Mathematical descriptions of the worst-case attacks are used to demonstrate the additional processing will provide similar advantages for cases for which we do not have numerical results. All the data-driven processing used in our approaches employ only unattacked training data.**

***Index Terms*—Cyber security, Sensor attack protection, powerful attack protection, connected vehicle networks, anomaly detection.**

I. Introduction

Incorporation of sensors into infrastructure provides important advantages [1]–[7]. The 2020 World Economic Forum's Global Risks Report listed cyber attacks on global critical infrastructure as a top concern that urgently needs to be mitigated [8]. Sensors are highly vulnerable to cyber attacks and cyber attacks on sensors can cause tremendous damage. Unfortunately, protection against such cyber attacks on sensors has not been adequately addressed [9]. Here we focus on protecting sensor-based sequence estimation/regression algorithms. These sensor-based sequence estimation/regression algorithms can be employed to estimate any quantity sensed by the sensors, for example: position, velocity and acceleration of objects of interest.

Many systems today depend on sensors. These include vehicles and vehicle networks, internet of things systems, and other smart systems. There are attacks on sensors that 1) give the attacker complete control over the attacked data; 2) allow attacks at a sensor to be changed arbitrarily vs time; 3) allow attacked sensors to be changed arbitrarily vs time; 4) Allow the number of sensors attacked to be changed arbitrarily vs time; 5) provide no prior knowledge on which sensors are more likely attacked. Mitigation approaches for such cases are lacking so we consider them here.

Consider a radar system as one example of an important sensor that can be attacked. During unattacked operation, as shown in Fig 1a, a radar transmits a pulsed wave in a given direction, which after bouncing off an object, maybe an airplane, is reflected back towards the radar. The pulsed wave is received at the radar with a given delay θ with respect to the transmitted waveform and with additive noise and clutter n. By knowing the speed of the wave c, the distance d from the radar to the object can be determined based on the delay. As shown in Fig 1b, a spoofing attack can be launched if an attacker receives the waveform transmitted by the radar, stores it in a digital memory for a while, and then plays back the waveform with an extra delay τ of its choosing. Since the attacker can choose the extra delay, he has complete control of the distance to the object that the radar reports, the sensor data.

Similar attacks apply to other active sensor systems[1], including lidar, sonar, and GPS. In GPS attacks, attackers often transmit fake GPS signals [10]. All such attacks are sometimes called in-band attacks since the attacks employ signals whose frequencies match those sensed by the sensor. Many other sensors allow similar attacks, including many of the in-band and out-of-band attacks described in [11]. These out-of-band attacks employ signals of vastly different frequencies from those sensed, impinging on sensors or connections to alter sensor data. The impinging attack signal modality can be acoustic, optical, or electromagnetic, while the sensor senses a different modality. One group of out-of-band attacks employ acoustic attacks targeting the resonant frequencies of gyroscopes and accelerometers. Many of these attacks, along with many classical spoofing and man-in-the-middle attacks, which

This work was sponsored by the Office of Naval Research (ONR) under grant number N00014-22-1-2626.

Rick S. Blum, and Xubin Fang are with the Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA 18015 USA (e-mails: rblum@eecs.lehigh.edu, xuf220@lehigh.edu).

Ramesh Bharadwaj is with the Naval Research Laboratory, Wahington DC USA (email: ramesh.bharadwaj@nrl.navy.mil).

Brian M. Sadler is with UT-Austin, Austin, TX USA (e-mail: Brian.sadler@ieee.org).

[1]Active sensor systems transmit signals.

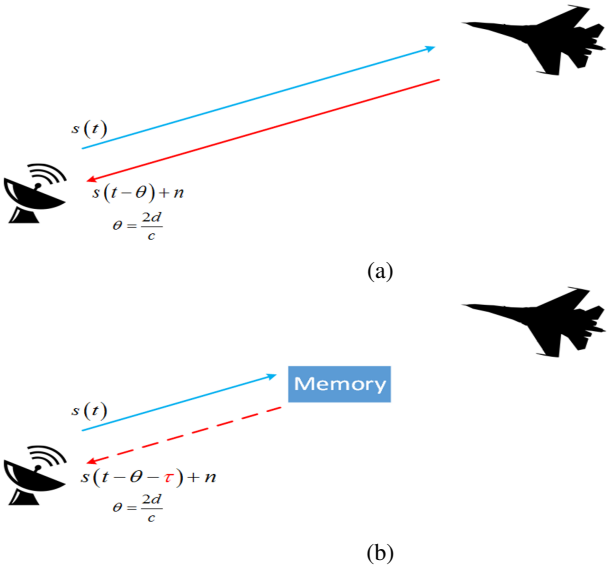


Fig. 1: Figure illustrating spoofing attack on a radar. (a) Unattacked. (b) Attacked.

modify transmissions of sensor data [12], allow attackers to replace actual sensor data with values of their choosing, a powerful type of attack we consider later.

A. Problem and Attack Model

This work aims to provide an outer shell of protection that can surround any unprotected sequence estimation/regression algorithm chosen from a large class. Without the shell, the original unprotected estimation algorithm is designed for cases without cyber attacks on the sensor data. The shell will eliminate problematic (in terms of degrading the estimate) attacked sensor data, allowing the protected estimation algorithm to operate using the remaining data. We focus on centralized sequence estimation algorithms in this paper, but we intend to later consider decentralized estimation algorithms in an attempt to show similar ideas can be employed there.

The attack model considered in this paper allows much more powerful attacks than we have ever seen successfully detected/eliminated in the existing attack mitigation literature. We assume the attacker has complete control to generate any sensor data values after the attack. We assume the protection system has no prior knowledge of which sensors are more likely attacked. We assume the attacked sensors and the attacks launched on those sensors can change each time step. We assume the protection system can't make any assumptions on the number of sensors attacked at any time step or about the patterns of attacks over time. Such attacks are consistent with sensor spoofing and out-of-band attacks discussed in Subsection I-A, where the signal impinging on the sensor is already modified and this modification changes a sensor output exactly as desired by the attacker. The attacked sensors can also exhibit such properties for other types of sensor attacks also, see Subsection I-A . We provide two protection approaches, one for attackers without detailed knowledge of the protection approach and another for attackers with detailed knowledge of our protection approach. We assume all the data-driven processing used in our approaches, for protection or for estimation, will employ only unattacked training data since it seems impossible to obtain training data accurately representing all possible attacks.

B. Contribution

This paper develops the first methods that accurately identify/eliminate only the problematic attacked sensor data (keep the rest) presented to a sequence estimation/regression algorithm under a proper powerful attack model that fits the situation based on known/observed attacks. Our approaches can be used as an outer shell of protection that can surround an unprotected estimation/regression algorithm (designed for unattacked sensor data), allowing the protected estimation algorithm to properly operate, with excellent performance, using the deemed to be unattacked data, which we have not seen for the attack model considered. Our approaches do not assume a known form for the statistical model of the sensor data, allowing data-driven and machine learning sequence estimation/regression algorithms to be protected. Our approaches employ only unattacked training data and handle the possible attacks allowed by the powerful attack model where the attacked sensors and attacks can change for each time step, the attacker has complete control of after-attack sensor data, and the protection system has no prior knowledge of which sensors are more likely attacked and how many sensors are attacked. Such powerful sensor attacks have not been mitigated elsewhere in the literature.

We initially focus in this paper on a simple protection approach for attacks not endowed with knowledge of the details of our protection approach, but later we describe additional processing for attackers with knowledge, thus allowing a lower complexity approach if suitable. The paper provides experimental demonstration of good performance for both the simple and additional processing. Our simple method is shown to achieve, to two decimal places, indistinguishable estimation performance in cases tested, which assume the attacker does not have knowledge of the protection scheme, in comparison to an optimized approach which knows which sensors were attacked. For cases where the attacker has knowledge of the protection approach, the additional processing can be configured so that the worst-case degradation under the additional processing and a large number of sensors attacked can be made significantly smaller than the worst-case degradation under the simple approach, and close to that for an approach which knows which sensors are attacked, for the same number of attacked sensors with just a slight degradation under no attacks. Guarding against the worst-case is extremely important and if this performance can be made acceptable, then performance will always be acceptable for any attacks. Since our method does not employ high-complexity searches over all possible subsets of sensors, it is a relatively low-complexity algorithm to reduce delay and needed hardware for practical implementation.

To accomplish the majority of what we have just described, we rely heavily on our discovery on how to mathematically

describe the worst-case attacks for our simple and additional processing approaches. This then allowed us to describe how to calculate the worst-case performance. We feel this is an important contribution of our work that can be employed to analyze other protection approaches in a similar manner. We hope others will employ these ideas in future work. While our numerical testing is limited, as all would be, the mathematical description of our worst-case attacks allow us to ensure, even for cases not tested, that proper choice of the additional processing parameters imply the worst-case degradation under the additional processing and a large number of sensors attacked can be made significantly smaller than the worst-case degradation for the simple approach for the same number of sensors attacked. Directly calculating the worst-case performance, by knowing the worst-case attack, is extremely efficient in reducing computations, which is especially important if you want to try many parameter settings. It avoids trying many different attacks, an approach others take. This would be difficult to use due to extremely high complexity and one would never obtain the actual worst-case attack performance.

C. Example Application

To provide a specific example application, we consider connected vehicle networks (CVNs). However the general ideas can be applied to other applications. In CVNs, sensor technology is being adopted by automotive manufacturers, creating distributed self-organized networks of many high-speed vehicles and infrastructure [13] where these vehicles can communicate with each other to share sensor data to make driving safer [14]. Using sensor data to help identify the time-varying position and velocity of objects of interest is an important fundamental building block in CVNs [15], [16] since it is used by most other required functions, such as navigation and collision avoidance [17]. It is well known that CVNs are vulnerable to sensor attacks [18]–[20] so this seems a good example application we can use as needed in the rest of the paper.

Section II provides a literature review. In Section III we describe our proposed methods and the worst-case attacks on these methods. Section IV presents our numerical results. In Section V we provide conclusions.

II. Literature Review

There has been study on protecting classical sensor-based sequence estimation algorithms, for example Kalman filtering approaches, which assume a known mathematical model (usually linear) for the sensor data, see [21]–[26] for example. As these protection methods exploit the known mathematical sensor data model, they are not applicable to protecting data-driven (machine learning) approaches where a known mathematical sensor data model is not available. While these methods are not applicable to the problem of interest here, it is worth noting that the methods of this type that achieve the best performance generally perform a search over all the possible subsets of sensors which could be attacked. Such searches greatly increase the implementation complexity and run time of these algorithms. Our goal is to avoid such searches to avoid these issues.

Another approach to the problem of inference with attacked sensor data has been studied under the topic of inference with Byzantine data [27], [28]. This work built on early work on the Byzantine Generals problem [29]. Unfortunately, these approaches are known to provide very poor performance for some attacks when the number of attacked sensors is unknown, which makes them unsuitable for our attack model. For example, it is common in these approaches to assume that more than half of the sensors are unattacked and to produce an estimate based on identifying, and then using, these sensors. This can yield very poor performance if the assumption is not true. Other assumptions lead to similar issues.

The Byzantine ideas have also been used in the distributed and federated learning paradigm [30], where processing nodes/agents share data and some nodes/agents launch attacks. In the distributed and federated learning paradigm, the agents may pass gradients as opposed to raw data, but the goal of rejecting the gradients representing attacked data, while keeping the gradients representing unattacked data, makes the problems similar in some sense. In the distributed and federated learning paradigm, computing the average of the unattacked gradients becomes very important and thus the robust computation of these averages, called robust aggregation, has also become important. One example of robust aggregation employs a median operation, which is robust to attacks unless the number of values (gradients) attacked is greater than half of the number of values aggregated, the number of sensors in our cases. Trimmed means/medians and related approaches (see [30]), which have also (along with the median) received attention in the signal processing community [31], are also applicable. Clearly these robust aggregation approaches are also applicable to our sensor attack problems. Unfortunately, for the attack model we consider these robust aggregation approaches can perform poorly for some attacks. In fact, [30]) states that none of these robust aggregation approaches can be guaranteed to give accurate aggregation results when the number of attacked sensors is unknown.

More recently, some distributed estimation approaches have been considered where the notion of the trust in an agent is available [32] in a multi-agent system. The trust describes prior knowledge, possibly from past interactions, on the likelihood that an agent is providing false or attacked information. It has been shown that when trust information is available, it is possible to significantly outperform the robust aggregation approaches [33], [34]. In some cases, one can even provide accurate estimates for cases where more than half of the agents are providing false/attacked data [32] if trust information is available. Unfortunately, the nature of the attacks considered here would not allow one to produce the required trust information. We simply do not have any information on which sensors are more likely to be attacked.

Recently, some machine learning-based encoder-decoder anomaly detection (EDAD) approaches were proposed for protecting general sequence estimation algorithms and significant progress on advancing the technology has taken place, see the full story in [35]–[37]. In particular, there have been a rapid

series of papers which have presented further improvement of the initial basic EDAD approaches employed. The approaches have progressed from using early technology, for example autoencoder technology, to much more sophisticated approaches, the latest being generative adversarial network technology. Using these anomaly detection approaches to protect against sensor attacks has several advantages: (1) No assumptions needed on the number of attacked sensors or amount of attacked sensor data. (2) We can take advantage of the great progress to incorporate the latest technology. (3) There are many available approaches that need only unattacked training data. Unfortunately, all of these EDAD approaches have a negative point which we discuss next. However, we have a method to augment these approaches that overcomes this negative point.

The EDAD methods learn a statistical model fitting all the possible (called valid) observed time sequences of unattacked sensor training data. The training data set is assumed to be sufficiently large to fully describe all unattacked sensor data. During anomaly detection, any sensor data not following that model will be marked as anomalous (either an attacked or broken sensor, we call both attacked here). Thus, the EDAD methods will identify data that is not consistent with the training data as anomalous. For example, if an observed sensor data sequence exhibits certain patterns in training data but different patterns during operation, an anomaly (attack) is detected. This enables very powerful verification that the sensed trajectories are possible under no anomaly.

On the other hand, these EDAD approaches have an issue for the trajectory state estimation problems we consider here which typically have more than one possible (valid) state sequence. To make things clear, consider the case where the state sequence being estimated is the position/velocity of some real object. There will be many possible (valid) state sequences in such an application but many of them do not follow the position/velocity of the real object we are tracking. If the attacker substitutes sensor data following one valid sequence/trajectory with sensor data following another valid sequence/trajectory, then this attack passes EDAD. This gives the attacker tremendous power to cause big problems. The attacker can lead the system to believe the state being estimated is following a very different trajectory than it is actually following, potentially inducing an extremely large error in the estimates based on this sensor data. This error can be as large as the error between two valid trajectories that are farthest apart, unbounded if this difference is unbounded. Similar problems occur for if we estimate the state of some machine or some some other state sequence.

To overcome this issue, after EDAD we employ novel additional checks that are shown to provide acceptably small degradations even with full knowledge attacks on a large number of sensors. The additional checks makes use of a machine learning predictor trained to predict a typical unattacked sensor's measurement of the actual state trajectory, for example the position/velocity of the real object we are tracking. Details are described later. We have not seen this approach used to fix issues with EDAD. In fact, we have not seen these issues discussed, even though EDAD has been suggested for state sequence estimation problems.

III. Proposed Methods and worst-case Attacks

Next, we describe our two approaches for augmenting EDAD to eliminate the issue described at the end of the previous section. Our approaches implement what we call an actual path consistency check (APCC) to test if the sensor data is following the actual trajectory of the state. The first approach, called APCC-SIMPLE, is designed for cases where the attacker does not have knowledge about the details of the protection approach. The second approach, called APCC-ADDITIONAL, is designed for cases when the attacker has knowledge about the protection approach. This second approach adds some additional checks.

First, we group all sensors such that each group contains all processed sensor outputs that predict the same component of the tracked trajectory. For the group of sensors predicting the first component of the trajectory (maybe the X component of an object's position as in the top left of Fig 2), all the sensors passing EDAD (AD in Fig 2) are put through our APCC, which attempts to check if the sensor data is following the path of the actual trajectory. This same processing is carried out for each group of sensors predicting all the other components of the trajectory. Then all the sensor data that passes this check, deemed to be unattacked sensor data, will be sent on to the estimation/fusion processing which will combine (fuse) this data, possibly with other data (including trajectory estimates at previous time steps), to produce the trajectory estimate. The estimation/fusion processing can be thought of as a possibly user-selected estimation algorithm that assumes all the sensor data input to it is unattacked.

The overall approach is illustrated in the block diagram in Fig 2. It should be noted that the APCC block includes the two options just discussed. The first option, called APCC-SIMPLE, consists of just the part labeled 1 in the APCC block in Fig 2. The second option, called APCC-ADDITIONAL, includes both the parts labeled 1 and 2 (2 builds on 1) in the APCC block in Fig 2. Next, we describe the details of the APCC block, under two subsections entitled APCC-SIMPLE and APCC-ADDITIONAL. This is followed by a subsection on the worst-case attacks and a subsection providing the details of the estimation/fusion processing.

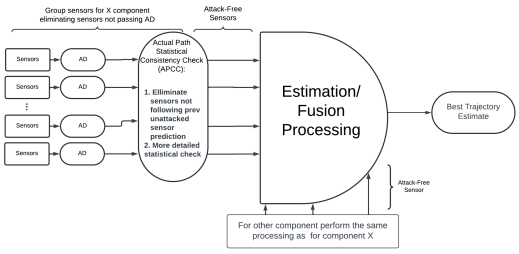


Fig. 2: Block Diagram of our approach. The block AD is an EDAD approach as per the text. We have two APCC options, APCC-SIMPLE (1. only) and APCC-ADDITIONAL (both 1 and 2) in the APCC block in the figure.

A. APCC-SIMPLE

Let us remind the reader that this version of the APCC is employed alone for cases where the attacker has no knowledge of the protection approach. For attackers with knowledge, we augment this approach as described in the next subsection. For APCC-SIMPLE, take any sensor measurements in a given group that have passed EDAD at a given time step and immediately subtract from them a prediction of what a typical unattacked sensor output would look like at that time step. The prediction comes from a trained machine learning algorithm whose input is sensor data from previous time steps which we deemed to be unattacked and possibly predictions at previous times. After the subtraction, we check that the difference lies in an interval chosen to contain a certain percentage β of all the possible values observed when computing this difference using all the unattacked training data. We refer to this percentage β as the consistency percentage. In some numerical results we select β to be 99.9%. Then any sensors producing differences not falling within the interval will be deemed attacked.

The prediction of what a typical sensor output would look like at a given time step can employ any machine learning approach. In the numerical results, we focus on a prediction method based on the Random Forest approach as delineated by Breiman [40]. Since the number of sensors labeled as unattacked can vary over time, we employ multiple Random Forest models to accommodate different numbers of input features. Accordingly, when the number of unattacked sensors changes, a corresponding Random Forest model with the exact number of unattacked sensor inputs is selected to perform the prediction. Once the predictor is trained, the processing in APCC-SIMPLE is illustrated in Fig 3 for $\beta = 99\%$.

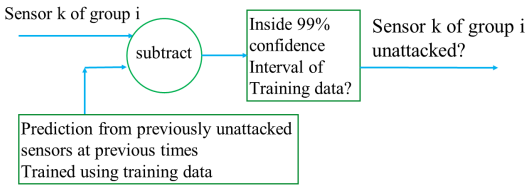


Fig. 3: Block Diagram of APCC-SIMPLE with $\beta = 99\%$.

B. APCC-ADDITIONAL

For attackers that have knowledge of our protection approach, we augment the APCC-SIMPLE approach with additional processing in an approach we call APCC-ADDITIONAL. Note that APCC-SIMPLE has already eliminated some sensors deemed to be attacked. This additional processing may eliminate more. To motivate the need for additional processing, consider the following. Suppose the malicious attackers know the details of the APCC-SIMPLE algorithm implemented to protect the sensor data. In that case, it's reasonable to infer that the attackers might insert many attacks (called edge attacks later) just inside one edge of the confidence interval shown in Fig 3 to avoid the attacks from being identified while causing maximum damage. Thus the edge attacks are the worst-case attacks on APCC-SIMPLE. Since these attacks will not be identified without additional processing, they will skew the estimation of the trajectory. If the attacker tries to launch a large number of such attacks, our additional processing, a data-driven approach, will attempt to eliminate many (or all) of these attacks by recognizing that they are not consistent with the statistics implied by the unattacked training data. Our approach will also mitigate other attacks.

For additional protection, we employ a different method to ensure the deemed to be unattacked data is statistically close to the training data, assumed unattacked, when closeness is measured in a different manner. The approach makes a decision on if a previously (based on APCC-SIMPLE) deemed unattacked sensor is attacked based on constructing the histogram of all the differences computed in Fig 3, at a given time and sensor group, that pass the APCC-SIMPLE test. We call this histogram $\hat{f}(x)$ when evaluated at a given bin x. Note that a histogram is often called an empirical probability density function (PDF) estimate and we use this to test if the statistics during operation match those obtained with training data in a specific sense.

In particular, APCC-ADDITIONAL uses the training data to find the smallest upper limit of a confidence interval, called $U(x)$, such that with probability $0 < \alpha < 1$ (usually closer to 1) the histogram $\hat{f}(x)$ should lie below $U(x)$ as per

$$\Pr(\hat{f}(x) < U(x)) = \alpha. \tag{1}$$

After learning $U(x)$ from the training data, we effectively eliminate sensor data which would cause $\hat{f}(x)$ to exceed $U(x)$. In particular, for any small histogram bin where $\hat{f}(x) > U(x)$, we exclude the smallest number of sensors that produce values in that bin so afterwards $\hat{f}(x) \leq U(x)$. The remaining sensor measurements, all those not excluded by either this or the previous processing, will be incorporated into the estimation/fusion processing as per Fig. 4 and Fig. 2.

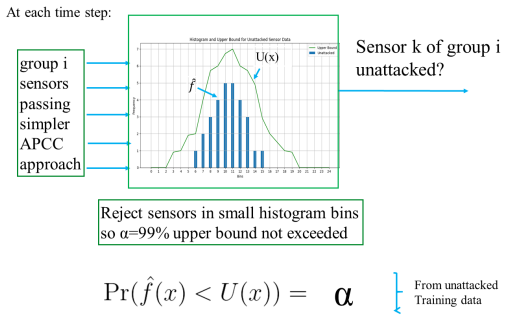


Fig. 4: Block diagram with $\alpha = 99\%$ illustrating how the APCC-ADDITIONAL processing builds on the APCC-SIMPLE processing, resulting in a decision on if each sensor is unattacked at a given time.

C. Worst-case Attacks

An important point is that we can calculate both the worst-case attack performance under the simple processing in APCC-SIMPLE (the edge attack mentioned previously

in Subsection III-B) and the worst-case attack performance under the additional processing in APCC-ADDITIONAL, a major contribution of our work. We will use this later in the numerical results to show the worst-case attack under APCC-ADDITIONAL can be made to cause significantly less damage that the worst-case attack under APCC-SIMPLE for cases with a large number of sensors being attacked, with the proper choice of the parameters in APCC-ADDITIONAL. If the worst-case attack can be tolerated, any attack can be tolerated.

At any given time step, the optimum N_A-sensor attack against the APCC-ADDITIONAL approach will try to insert N_A attacks whose magnitudes lie in a set of histogram bins[2] which each provide a large enough difference between the upper bound $U(x)$ and the sensor data histogram $\hat{f}(x)$ to support those attacks (so the attacks get through) while choosing those bins that cause the greatest degradation to the estimation/fusion. Such attacks are called water-filling attacks since one can imagine pouring water to fill up the space left between the upper bound and the sensor data histogram, but the water must be poured in the most damaging bins first.

The approach is illustrated in Fig. 5. The red lines show bins deemed to cause the most degradation to the estimation/fusion of all bins where there is enough room between the upper bound and the sensor data histogram to insert one or more attacks. As shown in Fig. 5, one can only add attacks to a given histogram bin if there is enough space between the upper bound and the sensor data histogram. This greatly limits the possible attacks which will pass this protection approach and the damage they can impose. Notice that increasing the number of sensors attacked beyond a certain value creates a smaller change in the possible degradation of the estimation/fusion for each additional sensor attacked since the additional sensors attacked have to eventually occupy bins which cause smaller degradation. At some point, an additional attacked sensor can no longer degrade the estimation/fusion more than typical noise at that sensor.

To find the particular N_A sensors in which we will launch those attacks, we should pick those sensors that lead to the greatest damage, called nonrandom attacks. Intuitively, these sensors will have unattacked sensor values that differ the most from the attack values. Thus if the attack will be launched on the far left-hand side of the x axis as in Fig. 5 (see red bins), then you should pick a sensor whose unattacked value is the farthest away, on the right-hand side of the x axis in Fig. 5 (away from the red bins).

D. Fusion Processing Block

At each time step, the possibly user-selected estimation/fusion processing will take all the sensor outputs passing the APCC and combine them with internally stored values to produce a fused scalar output for each trajectory component to be estimated. For example, the estimation/fusion processing can employ a set of previous predictions that can be saved internally inside the estimation/fusion block in Fig. 2.

[2]Bins are assumed small so we insert attacks in the middle of the bins.

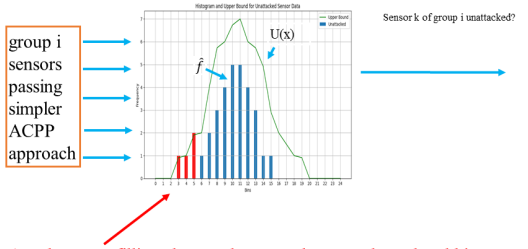


Fig. 5: Block Diagram illustrating the worst-case water-filling attack for APCC-ADDITIONAL. The red lines show places where there is enough room between the upper bound and the sensor data histogram to insert one or two attacks and these attacks cause the greatest degradation to the estimation/fusion.

The number of sensors deemed unattacked at a given time can change from time to time. This can be handled by choosing from several different machine-learning (or other) models, each with a different number of inputs. Specific examples are described in Section IV.

IV. Experimental Results

First, we remind the reader that, as explained at the end of Section II, EDAD approaches alone will give very large, potentially unbounded, errors for typical sequence estimation problems. We skip numerical results of this type to save space. By definition, by adding at least APCC-SIMPL to the EDAD approaches, we can limit the errors to reasonably small values of our choosing.

As CVN data sources with a sufficient number of sensors are unavailable and to test in a fully controllable environment, we generate sensor data used for experiments from a vehicle trajectory simulated using the Simulation of Urban Mobility open source software (SUMO) and introduce controllable sensor noise, independent samples from time to time. Most experiments will be conducted for cases of both Gaussian and Laplacian sensor noise, two very different noise models [38]. Let the scale parameter of the Gaussian noise be σ so that its variance is σ^2. Let the scale parameter of the Laplace noise be b so that its variance is $2b^2$ [38]. We vary the noise variance to characterize performance for different noise levels. Here we estimate a trajectory for a scalar quantity for simplicity, where the estimation/fusion algorithm uses only the sensor data at a given time step to produce the estimate at that same time step in the trajectory (no stored data) although we have considered other cases. We first employ SUMO to generate many different possible noise-free vehicle path trajectories of length $m = 150$ time steps that follow routes on a map shown in Fig 6. After adding independent noise from sensor to sensor to these trajectories, we have our sensor data. This sensor data will be used differently for testing APCC-SIMPLE and APCC-ADDITIONAL as discussed later.

While our protection approach allows changing the number of sensors attacked each time step, here we fix the number of sensors attacked each time step for ease of interpretation of the results and call the number of attacked sensors N_A. Here

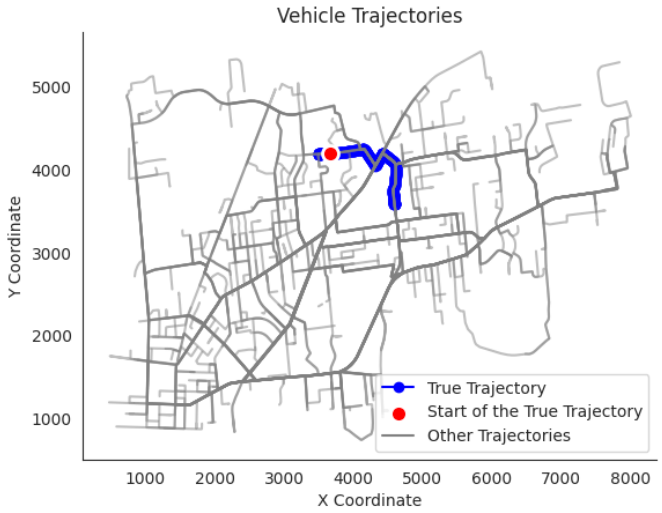
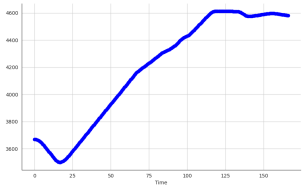
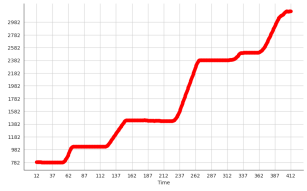


Fig. 6: Map used with SUMO to produce trajectories.



(a) x coordinate waveform of the true trajectory.



(b) x coordinate waveform of an attacked trajectory.

Fig. 7: Comparison of true and attacked trajectories.

$N_A < N$ where N is the total number of sensors available (attacked plus unattacked). In order to focus on the ability of our approach to find and eliminate attacked sensor data and to remove any effects due to the estimation/fusion algorithm in our numerical results, we employ Maximum Likelihood Estimation (MLE) [38], [39] as the fusion approach, which is optimized under the assumption of known sensor noise distribution. For Gaussian noise or a large number of provided sensor observations, the MLE is optimum among unbiased estimators. We have observed that these MLE results can well approximate performance for highly optimized machine learning-based fusion algorithms. We have also found that non-optimized machine learning-based fusion algorithms do impose some additional degradation in performance, while we do not provide these results here.

None of the attacks we consider in our tests would be detected by pure EDAD since these attacks substitute one valid sequence for another as discussed in the second to last paragraph of Section II. Our purpose here is to show our new approaches can achieve very good performance for such attacks, thus providing a useful way of augmenting EDAD to perform well for attacks that EDAD misses. We should note that for the kind of attacks we consider, it's obvious that using attacked data (rather than eliminating it) will never improve performance since these attacks delete all the useful information about the original trajectory by changing it.

Let the actual trajectory we wish to estimate be $y_1, \ldots, y_m$. Let the vector of sensor observations at time step $j, j = 1, \ldots, m$ be $X_j = (x_{1j}, \ldots, x_{Nj})^T$. The estimation/fusion algorithm will use this vector to produce an estimate for time step j which we denote as $\hat{y}_j$. For any given estimation/fusion method, the performance is measured using the mean square error (MSE), defined as

$$\text{MSE} = \mathbb{E}\left[\frac{1}{m}\sum_{j=1}^{m}(\hat{y}_j - y_j)^2\right]. \tag{2}$$

The Normalized Root MSE (NRMSE) can be defined as

$$\text{NRMSE} = \sqrt{\frac{\text{MSE}}{\frac{1}{m}\sum_{j=1}^{m}|y_j|}} = \sqrt{\frac{\mathbb{E}\left[\frac{1}{m}\sum_{j=1}^{m}(\hat{y}_j - y_j)^2\right]}{\frac{1}{m}\sum_{j=1}^{m}|y_j|}} \tag{3}$$

By employing an optimized estimate which also only uses the sensor data which is actually unattacked, we obtain a useful performance comparison that incorporates information not normally available in practice called the Genie Estimate (GE), The GE sets $\hat{y}_j$ in (2) as the MLE estimate/fusion for the known noise distribution using only the unattacked sensor data in X_j.

A. Numerical Results for APCC-SIMPLE: Attacks Without Knowledge

In this subsection, we discuss the numerical results obtained with the APCC-SIMPLE algorithm, described in Subsection III-A. In the experiments, the chosen β is 99.9%, the total number of time steps in the trajectory m is 150, and the sampling period of the sensor data is 1.0×10^{-3} sec. To obtain an accurate estimate of NRMSE in (3), we performed a Monte Carlo (MC) simulation where we average the NRMSE over 1 million independent realizations called the MC run length.

We consider cases where the attacker is assumed to not have detailed knowledge of the protection scheme, for example the confidence interval in Fig. 3 or the prediction. Thus the attacker is unable to launch optimum (worst case) attacks that are all focused to lie at the edge of the confidence interval in Fig. 3. We consider such cases later. Instead, for ease in comparing the various cases considered in this subsection, the noise-free unattacked trajectory was chosen to be the trajectory shown in Figure 7a, which is one of the trajectories from the set of those generated using the map in Fig 6. The noise-free attacked trajectory was chosen randomly (attacker can not optimize) from the other waveforms generated using the map in Fig 6. One of these possible attacked trajectories is shown in Figure 7b. The attacked trajectories are chosen randomly and independently at each attacked sensor without replacement.

In Table I and Table II, we show the numerical results for various cases we have tested. The numerical results illustrate that for these cases where we apply the APCC-SIMPLE approach with optimized MLE fusion to address attackers without protection system knowledge, the NRMSE between the APCC-SIMPLE (SIMPLE in Tables) estimated trajectory and the true trajectory (last column) is small and it is very close (identical to two decimal places) to that for the GE (using MLE). It indicates the APCC-SIMPLE approach successfully

identifies the attacked sensors and excludes their readings accordingly in the estimation/fusion process.

TABLE I: APCC-SIMPLE Gaussian Results $\beta = 99.9\%$
σ^2 is noise variance, N is # of sensors, N_A is # of attacked sensors, NRMSE GE and NRMSE SIMPLE (APCC-SIMPLE) both use MLE

σ^2	N	N_A	NRMSE GE	NRMSE SIMPLE
1.0×10^{-4}	50	40	5.03×10^{-5}	5.03×10^{-5}
1.0×10^{-4}	50	30	3.56×10^{-5}	3.56×10^{-5}
1.0×10^{-4}	50	10	2.52×10^{-5}	2.52×10^{-5}
1.0×10^{-4}	20	10	5.03×10^{-5}	5.03×10^{-5}
1.0×10^{-4}	10	5	7.12×10^{-5}	7.12×10^{-5}
1.0×10^{-2}	50	40	5.03×10^{-4}	5.03×10^{-4}
1.0×10^{-2}	50	30	3.56×10^{-4}	3.56×10^{-4}
1.0×10^{-2}	50	10	2.52×10^{-4}	2.52×10^{-4}
1.0×10^{-2}	20	10	5.03×10^{-4}	5.03×10^{-4}
1.0×10^{-2}	10	5	7.12×10^{-4}	7.12×10^{-4}

TABLE II: APCC-SIMPLE Laplacian Results $\beta = 99.9\%$
See TABLE I for column definitions.

$\sigma^2 = 2b^2$	N	N_A	NRMSE GE	NRMSE SIMPLE
2.0×10^{-4}	50	40	6.06×10^{-5}	6.06×10^{-5}
2.0×10^{-4}	50	30	4.11×10^{-5}	4.11×10^{-5}
2.0×10^{-4}	50	10	2.80×10^{-5}	2.80×10^{-5}
2.0×10^{-4}	20	10	6.06×10^{-5}	6.06×10^{-5}
2.0×10^{-4}	10	5	9.43×10^{-5}	9.43×10^{-5}
2.0×10^{-2}	50	40	6.06×10^{-4}	6.06×10^{-4}
2.0×10^{-2}	50	30	4.11×10^{-4}	4.11×10^{-4}
2.0×10^{-2}	50	10	2.80×10^{-4}	2.80×10^{-4}
2.0×10^{-2}	20	10	6.06×10^{-4}	6.06×10^{-4}
2.0×10^{-2}	10	5	9.43×10^{-4}	9.43×10^{-4}

B. Numerical Results for APCC-ADDITIONAL and Attacks With Knowledge

1) Edge Attack - APCC-SIMPLE vs APCC-ADDITIONAL : In this subsection, we consider the worst-case attack, called an edge attack, for the APCC-SIMPLE approach and demonstrate the performance improvements obtainable for such attacks when the APCC-ADDITIONAL approach is applied. The edge attack places the attacked sensor values just inside the edge of the confidence interval shown in Fig 3, thus they require knowledge of the protection approach to launch. In the experiments, the number of sensors N is fixed to be 50 while we vary the number of attacked sensors N_A to investigate the impact of attacking more sensors. The total number of time steps considered m in the trajectory to be estimated is still 150. To perform the APCC-ADDITIONAL processing, we use histograms of 25 bins to calculate $U(x)$ and $\hat{f}(x)$ in Fig. 4. We consider two specific types of edge attacks. One where we select the sensors to attack randomly, called a random attack, and the other where we select the sensors to attack which will cause the most damage, called a nonrandom attack. Since we pick the attacks near one edge of the confidence interval in Fig 3, the sensors which will cause the most damage are those with unattacked data closest to the other end of the confidence interval while lying inside the confidence interval.

For the case when the sensor noise is Gaussian, Table III (Non-Random attack) and Table IV (Random attack) show the NRMSE performance improvements of the APCC-ADDITIONAL approach over the APCC-SIMPLE approach for the edge attacks. For large N_A, the APCC-ADDITIONAL approach nearly perfectly identifies and removes all the edge attacked sensors to provide NRMSE performance close to the GE (identical to two decimal places). The performance of the APCC-ADDITIONAL approach is considerably better than that of the performance of the APCC-SIMPLE approach for the edge attacks which is extremely important in some critical applications. However, considering its simplicity. the performance of the APCC-SIMPLE approach for the edge attacks is not that bad and might be suitable in some noncritical applications. Note that the degradation of the worst-case attack (Edge attack) is bounded to a reasonably small value for APCC-SIMPLE but this will not be the case if we remove the APCC-SIMPLE processing as discussed previously. In the cases considered, the nonrandom attacks are more powerful than the corresponding random attacks as expected. Similar conclusions to those drawn from Table III and Table IV can be drawn from the cases shown in Table V (NonRandom attack) and Table VI (Random attack) which consider cases with Laplacian noise as opposed to Gaussian noise.

2) Water-filling Attack - Worst-case attack on APCC-ADDITIONAL processing: The previously considered edge attack is the worst-case (causes the most damage) attack for the APCC-SIMPLE processing, but the worst-case attack for the APCC-ADDITIONAL processing is the water-filling attack we discussed previously. In the water-filling attack considered in this subsection, we optimally select the sensors to attack, so we call this a nonrandom attack. Table VII illustrates the APCC-ADDITIONAL processing performance (NRMSE with MLE fusion) for several values of α and β for the nonrandom water-filling attack for some cases with Gaussian sensor noise. Note that while the majority of the results in Table VII are for water-filling attacks (worst-case for ADDITIONAL), the first row in Table VII below the headings is for edge attacks (worst-case for SIMPLE) to allow simple comparison.

The results in Table VII show that, with proper choice of the parameters α and β, we can find approaches that make the worst-case NRMSE performance of the APCC-ADDITIONAL processing with a large N_A smaller than that of the worst-case NRMSE performance of APCC-SIMPLE processing in the same case and close to GE. As Table VII illustrates, this decrease in worst-case NRMSE performance of the APCC-ADDITIONAL processing for cases with a large N_A comes with a small increase in the NRMSE for cases with $N_A = 0$ when compared with the APCC-SIMPLE approach in Table III (same as GE). Table VII shows that when the noise distribution is Gaussian with a variance of 1×10^{-4}, using $\alpha = 80\%$ and $\beta = 80\%$ yields worst-case performance with $N_A = 40$ that is within a factor of 2 of the GE for that case while yielding performance with $N_A = 0$ that is within a factor of 1.8 of the GE for that case. While this seems to provide good performance at either extreme, other choices of α and β allow different tradeoffs.

Table VIII illustrates similar findings for cases with Laplacian noise. Figure 8 illustrates the trade-offs that the different previously considered (α, β) approaches can achieve in terms of worst-case NRMSE performance (y-axis) for a large N_A versus the $N_A = 0$ NRMSE performance (x-axis) under

TABLE III: Gaussian Non-Random Edge Attack $\alpha = 90\%$ $\beta = 99.9\%$

Variance	approach	$N_A = 0$	$N_A = 10$	$N_A = 20$	$N_A = 30$	$N_A = 40$
1.0×10^{-4}	NRMSE GE	2.25×10^{-5}	5.93×10^{-5}	1.04×10^{-4}	1.54×10^{-4}	2.21×10^{-4}
	NRMSE APCC-SIMPLE	2.25×10^{-5}	2.10×10^{-4}	4.09×10^{-4}	6.12×10^{-4}	8.34×10^{-4}
	NRMSE APCC-ADDITIONAL	2.28×10^{-5}	6.65×10^{-5}	1.09×10^{-4}	1.56×10^{-4}	2.21×10^{-4}
1.0×10^{-2}	NRMSE GE	2.25×10^{-4}	5.93×10^{-4}	1.04×10^{-3}	1.54×10^{-3}	2.21×10^{-3}
	NRMSE APCC-SIMPLE	2.25×10^{-4}	2.06×10^{-3}	4.02×10^{-3}	6.02×10^{-3}	8.19×10^{-3}
	NRMSE APCC-ADDITIONAL	2.28×10^{-4}	6.08×10^{-4}	1.05×10^{-3}	1.54×10^{-3}	2.21×10^{-3}

Note: The NRMSEs in this table are obtained by using MLE as the fusion approach.

TABLE IV: Gaussian Random Edge Attack $\alpha = 90\%$ $\beta = 99.9\%$

Variance	approach	$N_A = 0$	$N_A = 10$	$N_A = 20$	$N_A = 30$	$N_A = 40$
1.0×10^{-4}	NRMSE GE	2.25×10^{-5}	2.52×10^{-5}	2.91×10^{-5}	3.56×10^{-5}	5.03×10^{-5}
	NRMSE APCC-SIMPLE	2.25×10^{-5}	1.44×10^{-4}	2.86×10^{-4}	4.29×10^{-4}	5.69×10^{-4}
	NRMSE APCC-ADDITIONAL	2.28×10^{-5}	2.52×10^{-5}	2.91×10^{-5}	3.56×10^{-5}	5.03×10^{-5}
1.0×10^{-2}	NRMSE GE	2.25×10^{-4}	2.52×10^{-4}	2.91×10^{-4}	3.56×10^{-4}	5.03×10^{-4}
	NRMSE APCC-SIMPLE	2.25×10^{-4}	1.43×10^{-3}	2.87×10^{-3}	4.28×10^{-3}	5.73×10^{-3}
	NRMSE APCC-ADDITIONAL	2.28×10^{-4}	2.52×10^{-4}	2.91×10^{-4}	3.56×10^{-4}	5.03×10^{-4}

Note: The NRMSEs in this table are obtained by using MLE as the fusion approach.

TABLE V: Laplacian Non-Random Edge Attack $\alpha = 90\%$ $\beta = 99.9\%$

Variance	approach	$N_A = 0$	$N_A = 10$	$N_A = 20$	$N_A = 30$	$N_A = 40$
2.0×10^{-4}	NRMSE GE	2.48×10^{-5}	4.62×10^{-5}	8.97×10^{-5}	1.54×10^{-4}	2.66×10^{-4}
	NRMSE APCC-SIMPLE	2.48×10^{-5}	8.88×10^{-5}	2.65×10^{-4}	1.55×10^{-3}	1.66×10^{-3}
	NRMSE APCC-ADDITIONAL	2.48×10^{-5}	4.85×10^{-5}	9.11×10^{-5}	1.54×10^{-4}	2.66×10^{-4}
2.0×10^{-2}	NRMSE GE	2.48×10^{-4}	4.62×10^{-4}	8.97×10^{-4}	1.54×10^{-3}	2.66×10^{-3}
	NRMSE APCC-SIMPLE	2.48×10^{-4}	8.90×10^{-4}	2.65×10^{-3}	1.52×10^{-2}	1.63×10^{-2}
	NRMSE APCC-ADDITIONAL	2.48×10^{-4}	4.78×10^{-4}	9.03×10^{-4}	1.54×10^{-3}	2.66×10^{-3}

Note: The NRMSEs in this table are obtained by using MLE as the fusion approach.

TABLE VI: Laplacian Random Edge Attack $\alpha = 90\%$ $\beta = 99.9\%$

Variance	approach	$N_A = 0$	$N_A = 10$	$N_A = 20$	$N_A = 30$	$N_A = 40$
2.0×10^{-4}	NRMSE GE	2.48×10^{-5}	2.80×10^{-5}	3.29×10^{-5}	4.11×10^{-5}	6.06×10^{-5}
	NRMSE APCC-SIMPLE	2.48×10^{-5}	5.76×10^{-5}	1.88×10^{-4}	1.35×10^{-3}	1.35×10^{-3}
	NRMSE APCC-ADDITIONAL	2.48×10^{-5}	2.80×10^{-5}	3.29×10^{-5}	4.11×10^{-5}	6.06×10^{-5}
2.0×10^{-2}	NRMSE GE	2.48×10^{-4}	2.80×10^{-4}	3.29×10^{-4}	4.11×10^{-4}	6.06×10^{-4}
	NRMSE APCC-SIMPLE	2.48×10^{-4}	5.73×10^{-4}	1.88×10^{-3}	1.36×10^{-2}	1.37×10^{-2}
	NRMSE APCC-ADDITIONAL	2.48×10^{-4}	2.80×10^{-4}	3.29×10^{-4}	4.11×10^{-4}	6.06×10^{-4}

Note: The NRMSEs in this table are obtained by using MLE as the fusion approach.

Gaussian sensor noise. In addition, Figure 9 shows the trade-offs in worst-case NRMSE performance (y-axis) for a large N_A compared to the no-attack NRMSE performance (x-axis) when subjected to Laplacian sensor noise.

While the just given results numerically indicate that a sufficient decrease in α will lead to a decrease in the worst-case performance with a large N_A, this can be deduced based on the construction of the APCC-ADDITIONAL approach. This follows since reducing α reduces the upper bound $U(x)$ which then eliminates the space between the upper bound and the histogram for all bins which can support one or more attacks. Thus the bins that have space for attacks to pass through become fewer as α is reduced. In fact, the most damaging bins of this type will typically be heavily impacted in this way first so that the attacks that pass the protection after α reduction will cause less damage to the estimation/fusion. It is also clear that increasing α will improve the performance under no attack since fewer sensors will be deemed to be attacked with other things equal.

It's important to note that any error in the prediction in Fig. 3 could negatively affect the performance of the proposed APCC approach. We omit results showing this since the idea is quite intuitive. An accurate prediction approach must be employed. One might be concerned that there could be an accumulation of prediction errors over time. Thus if some attacked sensors are determined to be unattacked at a given time step, these sensors will be used in the prediction for the next time step as described in Fig 3, possibly increasing the prediction error at the next step. This error could accumulate over time if the attacks always move the prediction in a consistent direction. However, experimental results show if we employ a good machine learning approach which is properly trained then by properly adjusting β and α in APCC-ADDITIONAL, the impact of even powerful optimized attacks always moving the prediction in a consistent direction do not cause large degradation over longer trajectories. Table IX shows results when the noise distribution is Gaussian with a variance of 1×10^{-4} for an attacker launching 40 optimum (non-random water-filling, consistent direction over time) attacks each time step on a total of 50 sensors. Table IX show that by setting α to 60% and β to 80%, the APCC-ADDITIONAL approach provides enough protection to counteract even this severe worst-case attack. The simulation results show that the NRMSE reaches 1.86×10^{-4} after 150 time steps. Even when extending the

TABLE VII: Gaussian Optimized Non-Random Water Filling Attack

Variance	approach	$N_A = 0$	$N_A = 10$	$N_A = 20$	$N_A = 30$	$N_A = 40$
1.0×10^{-4}	Worst-case for SIMPLE ($\beta = 99.9\%$)	–	2.10×10^{-4}	4.09×10^{-4}	6.12×10^{-4}	8.34×10^{-4}
	NRMSE for $\alpha = 80\%$, $\beta = 80\%$	4.15×10^{-5}	1.29×10^{-4}	2.56×10^{-4}	4.03×10^{-4}	4.47×10^{-4}
	NRMSE for $\alpha = 70\%$, $\beta = 80\%$	4.94×10^{-5}	1.29×10^{-4}	2.19×10^{-4}	2.60×10^{-4}	2.60×10^{-4}
	NRMSE for $\alpha = 60\%$, $\beta = 80\%$	5.24×10^{-5}	1.29×10^{-4}	1.86×10^{-4}	1.86×10^{-4}	1.86×10^{-4}
	NRMSE for $\alpha = 50\%$, $\beta = 80\%$	6.06×10^{-5}	1.22×10^{-4}	1.40×10^{-4}	1.40×10^{-4}	1.40×10^{-4}
	NRMSE for $\alpha = 40\%$, $\beta = 80\%$	7.14×10^{-5}	9.31×10^{-5}	9.33×10^{-5}	9.33×10^{-5}	9.33×10^{-5}
1.0×10^{-2}	Worst-case for SIMPLE (for $\beta = 99.9\%$)	–	2.06×10^{-3}	4.02×10^{-3}	6.02×10^{-3}	8.19×10^{-3}
	NRMSE for $\alpha = 80\%$, $\beta = 80\%$	2.45×10^{-4}	1.29×10^{-3}	2.74×10^{-3}	4.79×10^{-3}	5.53×10^{-3}
	NRMSE for $\alpha = 70\%$, $\beta = 80\%$	2.70×10^{-4}	1.29×10^{-3}	2.52×10^{-3}	3.12×10^{-3}	3.12×10^{-3}
	NRMSE for $\alpha = 60\%$, $\beta = 80\%$	2.91×10^{-4}	1.29×10^{-3}	1.82×10^{-3}	1.82×10^{-3}	1.82×10^{-3}
	NRMSE for $\alpha = 50\%$, $\beta = 80\%$	3.34×10^{-4}	9.17×10^{-4}	9.54×10^{-4}	9.48×10^{-4}	9.48×10^{-4}
	NRMSE for $\alpha = 40\%$, $\beta = 80\%$	3.91×10^{-4}	5.60×10^{-4}	5.60×10^{-4}	5.60×10^{-4}	5.60×10^{-4}

Note: The NRMSEs in this table are obtained by using MLE as the fusion approach.

simulation to 7.0×10^4 time steps, the NRMSE only increases to 2.80×10^{-4}, about 1.5 times larger. We observed similarly small increases for slightly larger α of 70% or 80%. These results demonstrate that with proper protection, the damage from attacks leaking through and degrading the prediction can be made small, even under the worst-case attack, allowing for acceptable performance.

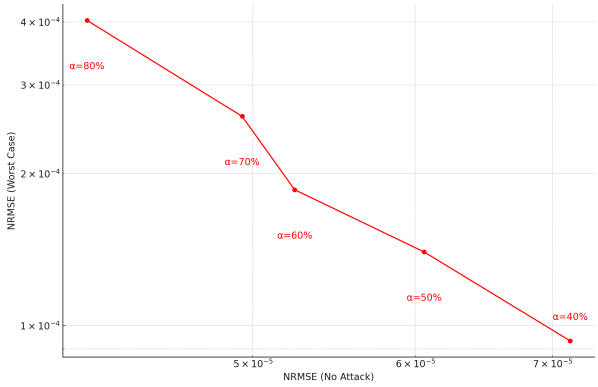


Fig. 8: Trade-off between worst-case NRMSE with $N_A = 40$ (Y-axis) versus NRMSE for $N_A = 0$ (X-axis) for a case with Gaussian sensor noise, various α, $N = 50, \beta = 80\%$, and noise variance = 1.0×10^{-4}.

V. Conclusions And Future Work

This paper developed the first methods that accurately identify/eliminate just the problematic attacked sensor data (keep the rest) presented to a sequence estimation/regression algorithm under a powerful attack model based on known/observed attacks. The developed approaches were shown to protect a sequence estimation/regression algorithm designed for unattacked sensor data by allowing such an algorithm to operate using the deemed to be unattacked data. The developed approaches employ only unattacked training data to mitigate all attacks allowed under the powerful considered attack model where the attacked sensors and attacks can change for each time step, the attacker has complete control of after-attack sensor data, and the protection system has no prior knowledge of how many sensors are attacked and which sensors are more likely to be attacked. Such powerful sensor attacks have not been mitigated elsewhere in the literature.

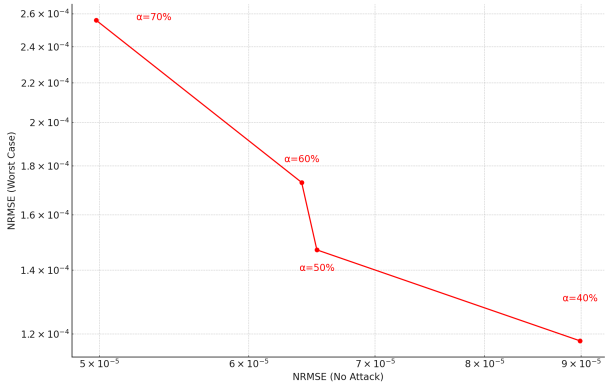


Fig. 9: Trade-off between worst-case NRMSE with $N_A = 40$ (Y-axis) versus NRMSE for $N_A = 0$ (X-axis) for a case with Laplacian sensor noise, various α, $N = 50, \beta = 60\%$, and noise variance = 2.0×10^{-4}.

We first proposed a simple protection approach for attacks not endowed with knowledge of the details of our protection approach, but later we proposed additional processing for attackers with knowledge. This allows a lower complexity approach to be employed if the attacker does not have detailed knowledge of the protection approach. Experimental results were provided which demonstrated good performance for both the simple and additional processing. Our simple method is shown to achieve estimation performance which is indistinguishable (to two decimal places) when compared to that of the GE (which knows which sensors were attacked) in cases tested, which assume the attacker does not have knowledge of the protection scheme. For cases where the attacker has knowledge of the protection approach, numerical results show that the additional processing can be configured, by proper parameter choice, so that the worst-case degradation under a large number of sensors attacked can be made significantly smaller than the worst-case degradation under the simple processing, and close to GE, for the same number of attacked sensors with just a slight degradation under no attacks. Guarding against the worst-case is extremely important and if this performance is acceptable, then performance will always be acceptable. Our protection approach is relatively low-complexity, as it does not employ high-complexity searches over all possible subsets of

TABLE VIII: Laplacian Optimized Non-Random Water Filling Attack

Variance	approach	$N_A = 0$	$N_A = 20$	$N_A = 30$	$N_A = 40$
2.0×10^{-4}	Worst-case for SIMPLE (for $\beta = 99.9\%$)	–	2.65×10^{-4}	1.55×10^{-3}	1.66×10^{-3}
	NRMSE for $\alpha = 70\%$, $\beta = 60\%$	4.98×10^{-5}	2.56×10^{-4}	2.56×10^{-4}	2.56×10^{-4}
	NRMSE for $\alpha = 60\%$, $\beta = 60\%$	6.40×10^{-5}	1.73×10^{-4}	1.73×10^{-4}	1.73×10^{-4}
	NRMSE for $\alpha = 50\%$, $\beta = 60\%$	6.52×10^{-5}	1.47×10^{-4}	1.47×10^{-4}	1.47×10^{-4}
	NRMSE for $\alpha = 40\%$, $\beta = 60\%$	8.99×10^{-5}	1.18×10^{-4}	1.18×10^{-4}	1.18×10^{-4}
2.0×10^{-2}	Worst-case for SIMPLE (for $\beta = 99.9\%$)	–	2.65×10^{-3}	1.52×10^{-2}	1.63×10^{-2}
	NRMSE for $\alpha = 70\%$, $\beta = 70\%$	4.02×10^{-4}	2.60×10^{-3}	2.60×10^{-3}	2.60×10^{-3}
	NRMSE for $\alpha = 60\%$, $\beta = 70\%$	4.70×10^{-4}	1.99×10^{-3}	1.99×10^{-3}	1.99×10^{-3}
	NRMSE for $\alpha = 50\%$, $\beta = 70\%$	5.50×10^{-4}	1.43×10^{-3}	1.43×10^{-3}	1.43×10^{-3}
	NRMSE for $\alpha = 40\%$, $\beta = 70\%$	6.60×10^{-4}	1.13×10^{-3}	1.13×10^{-3}	1.13×10^{-3}

Note: The NRMSEs in this table are obtained by using MLE as the fusion approach.

Parameter	**Setting or measurement**
Noise Distribution	Gaussian
Attack Type	Non-Random Water-Filling Attack
Variance	1×10^{-4}
α	60%
β	80%
N	50
N_A	40
NRMSE (150 Steps)	1.86×10^{-4}
NRMSE (70,000 Steps)	2.80×10^{-4}

TABLE IX: Comparison between NRMSEs for short run and long run

sensors.

We were able to mathematically describe the worst-case attacks for our simple and additional processing approaches, which allowed us to describe how to calculate the worst-case performance. The mathematical description of our worst-case attacks allowed us to show that proper choice of the additional processing parameters imply the worst-case degradation under the additional processing and a large number of sensors attacked can be made smaller, even for cases we did not test numerically. Since only a limited number of different cases can be numerically tested, this is important. We hope other researchers will employ similar ideas in their future work since these ideas seem very powerful. Directly calculating the worst-case performance, by knowing the worst-case attack is extremely efficient in reducing computations, which is especially important if you want to try many parameter settings. It avoids trying many different attacks, an approach others take. Such an approach would be difficult due to extremely high complexity and one would never get the actual worst-case attack performance.

We recognize that we have only taken the first steps in a new direction. There are still many more details that should be further investigated in the future and we list some of these in the rest of this section. The numerical results provided are extensive but are limited as any numerical results would be. It would be of interest to expand the cases tested in many ways. For example, in the presented numerical results, we focus solely on scalar trajectories, but in other investigations, not reported here, we tested the proposed algorithm on some higher-dimensional cases as well. We found the algorithm provides similarly good performance, compared to the cases reported here. Regardless, further testing would be desirable. As another example, we tested for a few specific statistical models, but we could expand the statistical models for the sensor observations to include many more models. It would also be of great interest to test using measured data. While we did not find suitable measured data to date, there are discussions of future initiatives which could provide this data in the future.

It would be of interest to further study methods to ensure good predictions where predictions are used in our approaches and maybe to monitor these predictions to make sure nothing has gone wrong. One issue of concern is to make sure problematic attacks do not leak through our protection to impact our predictions. Fortunately, we have found that this can be ensured even under very aggressive attacks if the APCC-ADDITIONAL approach employs α and β which provide the needed level of protection. On the other hand, it would be of great interest to carefully study methods which might further enhance performance on these cases. This seems possible.

In closing, it should be noted that our approach can be considered one level of protection which can be combined with other levels to enhance overall protection. As our approach focuses on difficult attacks, it would be interesting to understand the gains of combining multiple approaches in this manner.

REFERENCES

[1] P. K. Varshney, "Distributed Detection and Data Fusion", Springer Science & Business Media 2012.

[2] Z. Wan, W. Liu and P. Willett, "Non-Coherent Source Localization with Distributed Sensor Array Networks," 2022 IEEE 12th Sensor Array and Multichannel Signal Processing Workshop (SAM), Trondheim, Norway, 2022, pp. 86-90, doi: 10.1109/SAM53842.2022.9827843.

[3] B. Chen et al., "Heterogeneous Sensor Fusion With Out Of Sync Data," 2020 IEEE Aerospace Conference, Big Sky, MT, USA, 2020, pp. 1-6, doi: 10.1109/AERO47225.2020.9172681.

[4] R. Niu and P. K. Varshney, "Target Location Estimation in Sensor Networks With Quantized Data," in IEEE Transactions on Signal Processing, vol. 54, no. 12, pp. 4519-4528, Dec. 2006, doi: 10.1109/TSP.2006.882082.

[5] R. Viswanathan, Data Fusion. In: Computer Vision. Springer, Cham, https://doi.org/10.1007/978-3-030-03243-2_298-1, 2020.

[6] L. M. Kaplan, "Local node selection for localization in a distributed sensor network," in IEEE Transactions on Aerospace and Electronic Systems, vol. 42, no. 1, pp. 136-146, Jan. 2006, doi: 10.1109/TAES.2006.1603410.

[7] R Rajamäki, V Koivunen, Sparse Sensor Arrays for Active Sensing: Models, Configurations, and Applications, Sparse Arrays for Radar, Sonar, and Communications, 2024.

[8] The Global Risks Report 2020 — World Economic Forum (weforum.org)

[9] D. A. Gritzalis, G. Pantziou, R. Román-Castro, Sensors Cybersecurity. Sensors (Basel). 2021 Mar 4;21(5):1762. doi: 10.3390/s21051762. PMID: 33806381; PMCID: PMC7961485.

[10] P. Pradhan, K. Nagananda, P. Venkitasubramanium, S.Kishore and R.S. Blum, GPS Spoofing Attack Characterization and Detection in Smart Grids, IEEE Conference on Communications and Network Security, 2016.

[11] I. Giechaskiel and K. Rasmussen, "Taxonomy and Challenges of Out-of-Band Signal Injection Attacks and Defenses," in IEEE Communications Surveys & Tutorials, vol. 22, no. 1, pp. 645-670, Firstquarter 2020, doi: 10.1109/COMST.2019.2952858.

[12] Jiangfan Zhang, Rick S. Blum and H. Vincent Poor, "Approaches to Secure Inference in the Internet of Things", IEEE Signal Processing Magazine, Volume 35, Issue 5, pp. 50-63, 2018.

[13] A. J. Maidamwar and R. Sadakale, "Comprehensive study for localization techniques in manet and vanet," 2018 International Conference On Advances in Communication and Computing Technology, 2018, pp. 349–352.

[14] F. Qu, Z. Wu, F. Wang, and W. Cho, "A security and privacy review of vanets," IEEE T-ITS, vol. 16, no. 6, pp. 2985–2996, 2015.

[15] G. Soatti, M. Nicoli, N. Garcia, B. Denis, R. Raulefs, and H.Wymeersch, "Implicit cooperative positioning in vehicular networks," IEEE T-ITS, vol. 19, no. 12, pp. 3964–3980, 2018.

[16] H. S. Ramos, A. Boukerche, R. W. Pazzi, A. C. Frery, and A. A. F. Loureiro, "Cooperative target tracking in vehicular sensor networks," IEEE Wireless Communications, vol. 19, no. 5, pp. 66–73, Oct 2012.

[17] A. Boukerche, H. de Oliveira, E. Nakamura, and A. Loureiro, "Vehicular ad hoc networks: A new challenge for localization-based systems," Computer Communications, vol. 31, pp. 2838–2849, 2008.

[18] Yoshiyasu Takefuj, Connected Vehicle Security Vulnerabilities [Commentary], IEEE Technology and Society Magazine, pp. 15-18, March 2018.

[19] X. Sun, F. R. Yu and P. Zhang, "A Survey on Cyber-Security of Connected and Autonomous Vehicles (CAVs)," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 6240-6259, July 2022, doi: 10.1109/TITS.2021.3085297.

[20] Z. El-Rewini, K. Sadatsharan, N. Sugunaraj, D. F. Selvaraj, S. J. Plathottam and P. Ranganathan, "Cybersecurity Attacks in Vehicular Sensors," in IEEE Sensors Journal, vol. 20, no. 22, pp. 13752-13767, 15 Nov.15, 2020, doi: 10.1109/JSEN.2020.3004275.

[21] A.-Y. Lu and G.-H. Yang, "Secure state estimation for cyber-physical systems under sparse sensor attacks via a switched luenberger observer," Information sciences, vol. 417, pp. 454–464, 2017.

[22] Y. Shoukry, M. Chong, M. Wakaiki, P. Nuzzo, A. Sangiovanni- Vincentelli, S. A. Seshia, J. P. Hespanha, and P. Tabuada, "Smt-based observer design for cyber-physical systems under sensor attacks," ACM T-CPS, vol. 2, no. 1, pp. 1–27, 2018.

[23] S. Mishra, Y. Shoukry, N. Karamchandani, S. Diggavi, and P. Tabuada, "Secure state estimation: Optimal guarantees against sensor attacks in the presence of noise," in 2015 IEEE ISIT. IEEE, 2015, pp. 2929–2933.

[24] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," IEEE T-AC, vol. 58, no. 11, pp. 2715–2729, 2013.

[25] X.-J. Li and X.-Y. Shen, "A data-driven attack detection approach for dc servo motor systems based on mixed optimization strategy," IEEE Transactions on Industrial Informatics, vol. 16, no. 9, pp. 5806–5813, 2019.

[26] Z. Wang and Rick S Blum, "Algorithms and Analysis for Optimizing the Tracking Performance of Cyber Attacked Sensor-Equipped Connected Vehicle Networks", IEEE Transactions on Information Forensics and Security, vol. 16, pp. 5061 - 5076 Oct. 2021.

[27] A. Vempaty, L. Tong and P. K. Varshney, "Distributed Inference with Byzantine Data: State-of-the-Art Review on Data Falsification Attacks," in IEEE Signal Processing Magazine, vol. 30, no. 5, pp. 65-75, Sept. 2013, doi: 10.1109/MSP.2013.2262116.

[28] A. Vempaty, O. Ozdemir, K. Agrawal, H. Chen and P. K. Varshney, "Localization in Wireless Sensor Networks: Byzantines and Mitigation Techniques," in IEEE Transactions on Signal Processing, vol. 61, no. 6, pp. 1495-1508, March15, 2013, doi: 10.1109/TSP.2012.2236325.

[29] D. Dolev, "The Byzantine generals strike again," Journal of Algorithms, vol. 3, no. 1, pp. 14–30, 1982

[30] Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafael Pinot and John Stephan, "Byzantine Machine Learning Made Easy by Resilient Averaging of Momentums", ACM Computing Surveys, Vol. 56, No. 7, Article 169. Publication date: April 2024.

[31] Bovik, A.C., Acton, S.T. (1996). The Impact of Order Statistics on Signal Processing. In: Nagaraja, H.N., Sen, P.K., Morrison, D.F. (eds) Statistical Theory and Applications. Springer, New York, NY.

[32] Stephanie Gil, Michal Yemini, Arsenia Chorti, Angelia Nedić, H. Vincent Poor, Andrea J. Goldsmith, "How Physicality Enables Trust: A New Era of Trust-Centered Cyberphysical Systems", arXiv:2311.07492.

[33] M. Yemini, A. Nedić, A. J. Goldsmith, and S. Gil, "Characterizing trust and resilience in distributed consensus for cyberphysical systems," IEEE Transactions on Robotics, vol. 38, no. 1, pp. 71–91, 2022.

[34] M. Yemini, A. Nedić, S. Gil, and A. J. Goldsmith, "Resilience to malicious activity in distributed optimization for cyberphysical systems," in Proceedings of the 2022 IEEE 61st Conference on Decision and Control (CDC), 2022, pp. 4185–4192.

[35] X. Xia, X. Pan, N. Li, X. He, L. Ma, X. Zhang, and N. Ding, GAN-based anomaly detection: A review, Neurocomputing 493 (2022) 497–535.

[36] S. Noor, S. U. Bazai, M. I. Ghafoor, S. Marjan, S. Akram and F. Ali, "Generative Adversarial Networks for Anomaly Detection: A Systematic Literature Review," 2023 4th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2023, pp. 1-6, doi: 10.1109iCoMET57998.2023.10099175.

[37] Asif Ahmed Neloy and Maxime Turgeon, A comprehensive study of auto-encoders for anomaly detection: Efficiency and trade-offs, Machine Learning with Applications, Volume 17, 2024, 100572, ISSN 2666-8270,

[38] H. V. Poor, "An Introduction to Signal Detection and Estimation," 2nd ed., Springer, 1994.

[39] Steven M. Kay, "Fundamentals of Statistical Signal Processing: Estimation Theory", Prentice-Hall PTR, 1993.

[40] Leo Breiman, "Random Forests", Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[41] A. Dvoretzky, J. Kiefer, and J. Wolfowitz, "Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator," *Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 642–669, 1956.