

This item is likely protected under Title 17 of the U.S. Copyright Law. Unless on a Creative Commons license, for uses protected by Copyright Law, contact the copyright holder or the author.

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

**Please provide feedback**

Please support the ScholarWorks@UMBC repository by emailing [scholarworks-group@umbc.edu](mailto:scholarworks-group@umbc.edu) and telling us what having access to this work means to you and why it's important to you. Thank you.

# Digital Twin Based Topology Fingerprinting for Detecting False Data Injection Attacks in Cyber-Physical Systems

Javad Bahrami, Mohammad Ebrahimabadi, Mohamed Younis, Naghmeh Karimi

Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC)

Email: {jbahram1, e127, younis, nkarimi}@umbc.edu

**Abstract**—A Cyber-Physical System (CPS) employs interconnected sensing and actuation modules and applies distributed control strategies. With the major advances in communication technology, the CPS design methodology is getting broadly adopted, including in safety and mission-critical applications. The incorporation of digital twins within a CPS facilitates localized decision-making by the individual control modules within the system in a timely manner without risking stability and performance. However, cyberattacks could be detrimental when false data is injected to degrade the accuracy of the underlying digital twins so that a CPS module takes non-optimal or even risky action that causes application failure. This paper proposes a novel approach for detecting such an attack scenario through a combination of a predictive data model and a topology fingerprinting scheme. Specifically, we employ a recurrent neural network (RNN) to predict the next state (data) for the individual modules and use it to reason about the periodic updates provided by these modules. Then, we apply a data-driven fingerprinting scheme that characterizes the inter-module interaction to infer and classify anomalies based on the module-provided data. The validation results using a dataset of a smart power grid application demonstrate the effectiveness of our approach.

## I. INTRODUCTION

A CPS refers to a control system that involves a network of multiple actuators and sensing modules. The CPS design methodology pursues distributed coordination where data is shared continually among the various modules so that the right actions are performed to sustain stability. In essence, a CPS supports autonomous local decision-making while striving for global optimality. Such a design has gained popularity in recent years due to the major advances in computational platforms and wireless communication; practically the latter has been instrumental for large-scale infrastructure applications since pursuing wired connections is too expensive to be practical. Prominent examples of CPS applications include smart power grids, and intelligent transportation systems [1]. **CPS Security:** The increased autonomy and distributed management of a CPS design allow applications to scale in terms of the scope and size. Yet, with the increased automation comes the risk of security vulnerability. While the networked sensing, computation, and actuation components improve functionality, they expose the system to cyberattacks [2]. Specially a malicious behavior of a component may degrade or even disrupt the operation of the whole CPS, and may cause catastrophic consequences. Real scenarios include the cyberattack against Ukraine's power plants in 2015, and the one targeting a Czech hospital during the COVID-19 pandemic that caused a shutdown of its IT network

and canceled surgeries. Although reliability-related component failure can be mitigated through redundancy, the adaptive and subtle nature of cyberattacks requires security provisioning at the CPS design level to safeguard the system. To illustrate, an attacker can simply inject false data during the inter-module status update to steer the system towards instability.

**Digital Twins:** The major recent advances in deep learning techniques have enabled the development of accurate data-driven models for processes that are either too complex to mathematically formulate or governed by unknown scientific laws. The expressiveness of these data-driven models has led to the introducing the concept of *digital twins* (DTs), where the behavior of a module can be captured. DTs find applications in many domains ranging from space to defense. In the context of CPS, employing DTs seems to be quite natural and advantageous since a module can predict changes in the environment and anticipates the behavior of other system components [3]–[5]. Specifically, the incorporation of DTs in the CPS design mitigates data exchange latency and hence improves the system's robustness and responsiveness. In this paper, we leverage DTs in detecting anomalies in CPS systems and further classify an anomaly as malicious or accidental.

**Contribution:** This paper considers a CPS that consists of  $N$  interconnected components. Each component  $C_i$ ,  $0 \leq i \leq N-1$ , has a digital twin,  $DT_j$  for each component  $C_j$ ,  $i \neq j$ , to facilitate timely local decision-making. Normally, data exchange among components is used to update the individual DTs; yet in our work such data is also used to detect and classify anomalies. Particularly, we consider data forgery attacks where an adversary tries to introduce false data so that wrong local decisions are made and the entire CPS becomes unstable. Such a false data injection (FDI) attack can be launched through means like impersonation, message replay, man-in-the-middle, or even through capturing and manipulating one of the CPS components. To detect FDI attacks, we propose a novel Digital twin enabled CPS Topology Fingerprinting (DiToF) approach. DiToF pursues a two-step solution strategy.

In DiToF, the first step opts to recognize inconsistency in the provided data using RNN. Specifically, a component  $C_i$  will deploy a long short-term memory (LSTM) model for each other  $C_j$  in the system. Such an LSTM serves as a DT and is used to predict the next data sample from the underlying component. To elaborate,  $C_i$  will include an LSTM that is trained offline based on data collected from  $C_j$ . During operation,  $C_i$  will use  $LSTM_j$  to predict the next state update

from  $C_j$  and will also be fed with the actual data to update its weights for future inference. The response of  $LSTM_j$  at time  $t$  will be compared with the actual  $C_j$  data to detect inconsistency. To reason about whether the consistency reflects an FDI attack, the deviations of  $C_j$  noted at all components are considered by a One-Class Support Vector Machine (OCSVM) to determine whether the deviation is accidental, e.g., due to an external event, or deliberate due to cyberattack. In other words, the OCSVM captures the interrelation among the components, i.e., the CPS topology, in detecting any malicious data manipulation associated with each CPS component.

DiToF is validated using a smart power grid application where a dataset of PMU readings are used to construct and train LSTMs. Events caused by external factors such as winds, and by data manipulation are then used to demonstrate the effectiveness of DiToF. In summary, the contributions are:

- Develop DiToF, an effective DT-based scheme to detect anomalies in CPS;
- Devise a data-driven mechanism to distinguish between accidental events and deliberate cyberattacks in CPS;
- Validate DiToF's effectiveness using a power grid dataset.

## II. RELATED WORK

Given the distributed-control nature of CPS, ensuring the reliability and integrity of DTs over the course of usage is highly crucial. However, little attention has been paid to the vulnerability that the incorporation of DTs introduces. Shen et al. [6] have proposed a sequential hypothesis testing to assess DT susceptibility to data integrity attacks in the context of micro power grids. In [7] Physically Unclonable Functions (PUFs) are employed to authenticate data updates and prevent DT degradation through FDI. Nonetheless, this technique is solely capable of ensuring the security of data transmission and does not mitigate the threat of compromised components. In this paper, we rely on an LSTM model to detect inconsistency between the component's DT and the provided data.

Detection of FDI attacks has received significant attention in recent years, especially for smart power grids. Manandhar et al. [8] apply a Kalman-filter based method to detect anomalous data. In [9] a combination of the k-means++ and EM algorithms are used to devise a model for error estimation. Deng

and Liang consider an adaptive attacker who evades detection by changing the injected false data based on some predictive measurements of the transmission bus [10]; a power flow redistribution strategy is pursued as a countermeasure. Some work uses AC-state estimation in detecting FDI, rather than analyzing DC data [11]–[14]. Some work employs machine-learning techniques to detect such attacks. For example, Wang et al. [15] deployed deep-learning to detect and localize FDI attacks. Federated learning is used in [16]. Yu et al. [17] train an RNN model using some frequency and time domain features. A semi-supervised learning model is proposed in [18] to distinguish between normal and malicious data. However, almost all previous work has been tailored for centralized control systems, and in many of them, the detection scheme is not generic, and is highly application-specific.

## III. SYSTEM MODEL AND PRELIMINARIES

### A. System and Threat Models

We consider a CPS comprising  $N$  components, internet-worked using wireless links. Each component broadcasts status updates periodically to the rest of the network. We assume that a single broadcast would reach all components, i.e., the components form a complete graph with direct links. The operational CPS model opts to achieve global system-wide objective through local monitoring and decision-making at the component level. The CPS architecture is illustrated in Fig. 1. To increase responsiveness and enable scalability to a large number of components, the CPS design leverages the DT technology where each component  $C_i$  includes the individual DTs of all other  $N - 1$  components. In other words,  $C_i$  incorporates  $DT_j$  for all  $j \neq i$  ( $0 \leq i, j \leq N - 1$ ) to predict the state of other components when making local decisions and engaging its actuators. The DT engagement avoids the system sensitivity to communication latency if the actual state update is to be used instead. The DT is to stay current by factoring in the actual data in updating the underlying predictive model.

DiToF opts to detect an adversary who could compromise a component,  $C_a$  and uses it to launch FDI attacks. The false data may be meant to: (i) cause instant system instability, or (ii) gradually introduce inconsistency to the system state to degrade the operation efficiency. In the context of the assumed system model, the second scenario would undermine the DT accuracy of  $C_a$  at one or multiple other CPS components. DiToF opts to detect FDI attacks by analyzing the discrepancy between the DT-predicted states and the actual shared data and factoring in the correlation among the states of CPS components. The scope of the attack is assumed to be confined to a single component, i.e., no collusion. The term “Event” refers to the cases when the CPS is exposed to an accidental (unintentional) stimulus, e.g., wind.

### B. Digital Twin

A DT refers to a software-based model replica of a physical object, process, or system that mimics the behavior of the original counterpart [19]. DTs are mainly deployed in cyber-physical systems to optimize the process of the distributed decision-making process in each node. Machine learning models are typically employed to realize such DTs based on the data collected from sensors, actuators, etc. Due to the dynamic

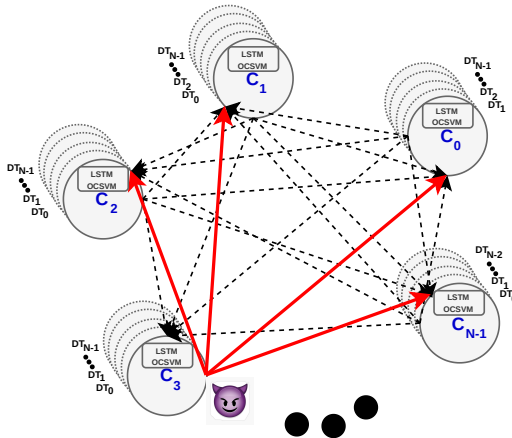


Fig. 1. An overview of the underlying CPS architecture containing  $N$  components, each includes the DTs of the other  $N-1$  components.

nature of status messages (data) within CPS, synchronization between each physical node and its twin is of utmost importance. Such synchronization followed by the update of DTs can be made in real time, thanks to reinforcement learning techniques. This update is required to enable an accurate distributed decision-making process in the CPS.

#### IV. FDI DETECTION METHODOLOGY

DiToF opts to detect malicious behavior expedited by attempts to inject false data. The detection methodology comprises two steps. The first leverages DTs to reason about the provided data from each component and points out anomalies. The second step classifies the detected anomaly as unintentional (due to an event) or malicious that caused by an attack. Such classification is conducted by correlating the data deviations of all components obtained in the first step to determine malicious behavior of any component reflected by an FDI attack. Such correlation simply captures the topological relationship among components, i.e., which pair of components are topologically dependent. We stress that such correlation is generic and is not application dependent; hence DiToF can be applied to any CPS. Before delving to the technical details of each step, we note that DiToF runs locally on the individual components with no centralized control.

##### A. Anomaly Detection

As shown in Fig. 1, we consider a CPS with  $N$  components; each component  $C_i$  includes the DT of the other components, referred to as  $DT_{i,j}$  where  $0 \leq i, j \leq N-1, i \neq j$ . In DiToF, each  $DT_{i,j}$  is realized as an LSTM model of  $C_j$  that is trained offline (and being updated in deterministic time frames) using the data sent from  $C_j$  to  $C_i$ . Since an LSTM has a memory cell that allows retaining and utilizing information from previous time steps, it can capture and remember patterns and context from the past. Therefore, an LSTM is a preferred choice in the case of CPS, as the data provided by the components, e.g., sensors, has a time series nature, i.e., typically changes with a specific sequence in time. During the CPS operation, the  $DT_{i,j}$  ( $LSTM_{i,j}$ ) predicts the next data value that  $C_i$  expects to receive from  $C_j$ . To detect anomalies, DiToF factors in the difference between the predicted value  $P_{i,j}(t)$  and the observed data  $O_{i,j}(t)$  (i.e., the actual data that  $C_j$  sends at time sample  $t$ ) over a time window of  $L$ . Specifically, DiToF calculates the Root Mean Square Error (RMSE) using:

$$RMSE_{i,j} = \sqrt{\frac{1}{L} \sum_{q=t-L+1}^t (O_{i,j}(q) - P_{i,j}(q))^2} \quad (1)$$

In Eq. (1),  $L$  represents the number of considered samples, and  $(O_{i,j}(t) - P_{i,j}(t))$  determines how far off the predicted values are from the corresponding real ones. RMSE is widely accepted due to its sensitivity to errors, mathematical convenience, and balancing of positive and negative errors [20]. The calculated RMSE is compared with a threshold value; staying below the threshold is deemed to be within a normal range, warranting no action. Otherwise, a data anomaly is assumed and component  $C_j$  is flagged by  $C_i$ . DiToF then takes another step to classify the anomaly as an accidental event or an attack, as we explain in the next subsection.

Finding an appropriate threshold is not straightforward. The threshold value would naturally depend on the CPS application, design parameters, and the characteristics of the underlying components. For example, in the case of power grids, these parameters can include the location of the Phasor Module Units (PMUs), seasonality, regional populations, etc. Therefore, DiToF determines the threshold value for each component based on the deviation of the predicted values from the observed ones (using Eq. 1) during the training process. Such training is conducted using data that does not include anomalies (accidental events or attack scenarios). Then, for every predicted sequence of length  $L$  during the inference, the RMSE value should be found again based on the inference data and compared with the threshold value that was calculated at model training time.

##### B. Anomaly Categorization: Accidental Event or Attack?

If a data anomaly is detected, i.e. an RMSE value of  $C_k$  exceeds its related threshold, the corresponding index ( $k$ ) is sent to our classification module realized via an OCSVM model. While the LSTM differentiates normal from abnormal cases, the OCSVM model categorizes the anomaly as an attack or accidental event. In DiToF, the OCSVM model conducts such classification in each component  $C_i$  based on the Pearson correlation between the inputs of the flagged component  $C_k$  and all other  $DT_{i,j}$ . For example, in the case of a CPS with 100 components, the Pearson correlation between the last  $L$  readings from  $C_k$  and the other 99 components is calculated and the results are used as features to the OCSVM, which in turn labels the anomaly of  $C_k$  as accidental or FDI attack. To the best of our knowledge, DiToF is the first method to use such a metric for security analysis.

Specifically, in each component  $C_i$ , the data received from the flagged component  $C_k$  is correlated with data provided by all the remaining components  $C_j$  ( $j \neq k$ ). As  $C_k$  has been flagged, the correlation engine starts receiving inputs, finding respective Pearson correlation coefficients (CORR), and forwarding them to the OCSVM module. The rationale is that inherently there is a functional relationship between the various components that will be affecting the system—whether accidental events or malicious attacks. Accordingly, we made use of this relationship to cluster the CORR values into two main categories, either accidental or malicious (referring to an attack). Also, note that the OCSVM module is in the hibernate mode during the normal operation of the system, and imposes no additional overhead. Although one could think of using an Artificial Neural Network (ANN) instead, we refrained from using such a model since:

- Unlike OCSVM, ANNs require data samples for all the considered classes; in practice the training dataset of CPS applications may not have attack data.
- A large dataset is required for training an ANN, and the training is computationally expensive.
- ANNs are typically large with multiple hidden layers and it could lead to an implementation challenge when the component  $C_i$  is memory-constrained.

Considering the aforementioned reasons we deploy a one-class clustering algorithm in this paper to differentiate accidental events and attacks from each other. Indeed, among

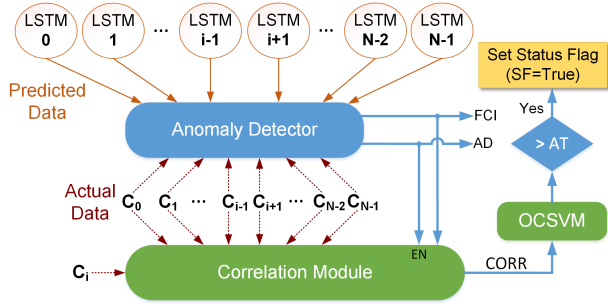


Fig. 2. Block diagram description of DiToF's operation on node  $C_i$ .

such one-class clustering schemes, OCSVM has been selected due to its low overhead and acceptable performance. The OCSVM is a machine-learning algorithm used for anomaly detection. Unlike traditional SVM variants that are designed for binary classification tasks, OCSVM focuses on identifying outliers or anomalies within a dataset. It works by creating a hyperplane that aims to encompass the majority of the data points (labels as +1), and data points lying outside this boundary are considered anomalies (labels as -1). Note that DiToF decides only based on the correlation between the flagged component,  $C_k$  and the other components; hence  $N-1$  CORR values are computed, including  $C_i$  which is the one that detected the anomaly.

### C. Summary of Steps

Fig. 2 shows a block diagram illustration of the various DiToF modules and their interaction. The LSTM cluster contains DTs of all other components. The anomaly detector module finds the RMSE values in real-time considering a predefined window length of  $L$  and raises the  $AD$  flag in case of any discrepancies beyond the predefined RMSE value between the predicted and observed values (recall Eq. 1). In such a case, the correlation module is informed about the suspected components index and then the OCSVM module is provided with the correlation results of the other components. Finally, the OCSVM output is compared with the threshold using the Comparator module (shown in blue in Fig. 2) resulting in an update of the Status Flag ( $SF$ ) if needed. Algorithm 1 presents the detailed steps of DiToF. The system's parameters get their initial values in lines 1-5. The LSTM and OCSVM go under their training/fitting phases in lines 6-12 (the training parameters are briefly explained in Section V). The optimum value of the attack threshold for the OCSVM module is found during the training in line 13 such that it results in the most accurate classification during inference. In lines 15-22, the system starts its normal operation (inference). While running the LSTM models, the component keeps track of the  $AD$  flag and once an anomaly happens ( $AD = True$ ), the  $EN$  flag gets updated in line 18, and the index of the flagged component ( $FCI$ ) gets forwarded to the correlation module. In the remaining lines, the correlation module finds the Pearson correlation between the  $C_k$  and all remaining  $N-1$  components ( $C_j$  where  $j \neq k$ ). The Status Flag ( $SF$ ) gets updated accordingly. All  $FCI$ ,  $AD$ , and  $SF$  values that relate to  $C_i$  are provided for further processing. The notations used in Algorithm 1 are summarized in Table I.

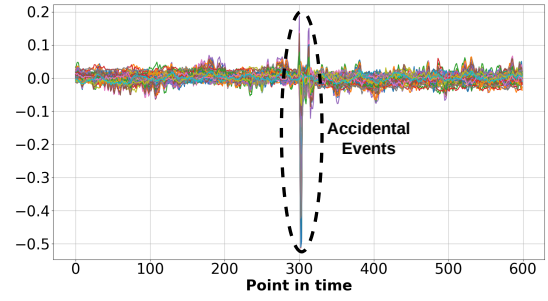


Fig. 3. The pmuBAGE [21] original normalized voltage values of 100 PMUs.

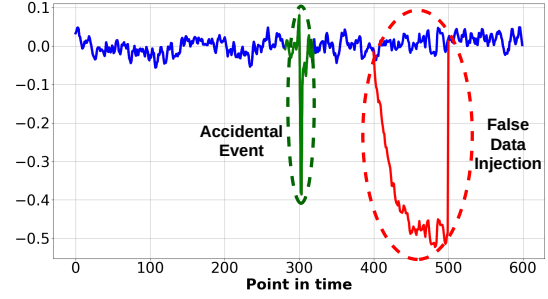


Fig. 4. Representation of normal (blue), accidental event (wind) in green, and malicious fault injected in red.

## V. VALIDATION RESULTS

### A. Experimental Setup

To validate DiToF, we have considered a smart power grid as an example of CPS and used the pmuBAGE dataset [21]. Such a dataset is based on a 20 sec time window with 30 samples/sec for a total of 600 data samples. This data is synthetic and based on collected samples from a real power plant; the values are normalized and shifted to sustain confidentiality. The effect of events, such as lightning strikes, line trips, wind, equipment failure, etc. is also captured. In total, it consists of  $PQVF$  data for 620 event-induced changes of voltage and 84 case resulting

#### Algorithm 1 : Pseudo Code of the DiToF

```

1: SF ← False                                     ▷ Initialization
2: node_count ← N
3: EN ← False
4: AT_List ← NULL
5: Corr_list ← [NULL*(node_count-1)]
6: for (j = 0, j < N, j++) do                      ▷ Start Training
7:   AT_j ← 0.0
8:   train_LSTM_Cj()
9:   AT_List.append(AT_j)                          ▷ Saving attack thresholds
10:  save.LSTM.state_dict.pt                       ▷ Saving model parameters
11:  fit_OCSVM_Cj()
12: end for
13: AT ← Optimum(AT_List)
14: load.LSTM.state_dict.pt                       ▷ Loading model parameters
15: while True do                                  ▷ Inference infinite loop
16:   {AD, FCI} ← test.LSTM()
17:   if AD then                                    ▷ Anomaly detected
18:     EN ← True
19:     k ← FCI                                     ▷ Saving FCI for future correlations
20:     break
21:   end if
22: end while
23: for (j = 0, j < N, j ≠ k, j++) do              ▷ Finding correlations
24:   Corr_list.append(CORR(C_k, C_j))
25: end for
26: if Predict_OCSVM_Cj(Corr_list) > AT then
27:   SF ← True                                     ▷ Updating SF
28: end if
29: return {SF, AD, FCI}

```

TABLE I  
NOTATION USED IN THE ALGORITHM 1

| Symbol | Description                              |
|--------|--|
| $SF$   | Status Flag (boolean)                    |
| $AT_i$ | Attack Threshold of $C_i$ (double)       |
| $AD$   | Anomaly Detected (boolean)               |
| $EN$   | Enable Correlation (boolean)             |
| $FCI$  | Flagged Component Index (integer)        |
| $CORR$ | Pearson Correlation Coefficient (double) |

frequency changes. However, we only used the voltage-related events in our validation, as plotted in Fig. 3. Our anomaly detection methodology can be applied to frequency-change events in a similar way.

The DT of every PMU is realized using an LSTM model trained initially offline and updated during operation (simulation). For each PMU, we also associate a correlation module along with an OCSVM engine. The PMU data is assumed to be exchanged through wireless broadcast. In the implementation, we assumed that PMU #3 in Fig. 1 acts maliciously and injects false data; recall that DiToF detects FDI attacks launched by a single component. Since the dataset does not capture cyberattacks, we have modeled FDI attacks consistent with the effect of accidental events (Fig. 4). We have limited the attack duration to around 3.5 sec to demonstrate the robustness of DiToF; in practice, FDI attacks could last significantly longer [22], which even improves their detectability by DiToF.

### B. Model Implementation and Training

For training/validation of the LSTM model reflecting the DT of a CPS component, we picked the initial 200 data samples (points in time) from the pmuBAGE dataset; out of which 180 samples were used for training, and the final 20 samples were used (as the future points) to validate the model and adjust the hyperparameters. After 100 epochs of LSTM training, we got a loss value in the order of  $\mathcal{O}(10^{-7})$  with a dynamic learning rate of the scheduler primitives (*ExponentialLR*) in Pytorch. The LSTM model is simple, employing just a single layer consisting of 50 neurons, and it is accompanied by a fully connected layer to capture temporal sequence patterns.

As shown in Fig. 4, in the pmuBAGE dataset an accidental event occurs at points 250 : 350 in time while the FDI attack occurs within the range 400 : 500. It took around 58 seconds to train a single LSTM model using Python V3.8.8 and Pytorch V1.11.0 on an 8-core 3 GHz CPU with 16GB RAM. The time complexity of training OCSVM is negligible compared to LSTM. We performed the test at 3 different times to cover all three cases of normal, accidental events, and attacks. In particular, based on the data shown in Fig. 4, points 250-350 were used to predict the following 50 points (representing an accidental event), points 400-500 were used to predict the following 50 points, representing both the attack and normal prediction of the LSTM. As we modeled the attack, we only switched it on during the evaluation of the model under an attack, otherwise, we turned it off.

### C. Experimental Results

The first set of results are shown in Figures 5-7. As mentioned earlier, these results relate to the cases where the LSTM is fed with 100 data points, depending on whether it experiences normal, accidental event, or attack conditions, and

it predicts 50 points into the future. As depicted in Fig. 5, in the normal operation mode, the LSTM model is performing well (blue curve), consistently generating predictions that closely match the actual data (green curve). The LSTM performance under an accidental event (wind) is shown in Fig. 6. While occasional discrepancies exist due to the event, they are generally within an acceptable range, signifying the model's ability to capture essential trends and patterns in the dataset. Finally, the performance of the LSTM under an FDI attack is shown in Fig. 7. To find the attack threshold (shown in green in Fig. 7), the initial 100 points of the dataset are used, which do not reflect any events or attacks. As expected, in this case, the LSTM model's performance falls below expectations, as it consistently produces predictions with significant errors when compared to the actual data. Note that such low performance in the case of attacks or events is in favor of DiToF as this reflects the differences between the predicted and actual values by the LSTM models and results in detecting anomalies.

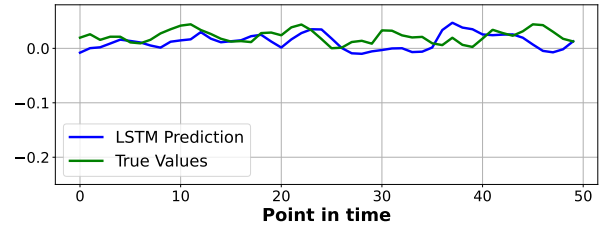


Fig. 5. LSTM performance under normal condition.

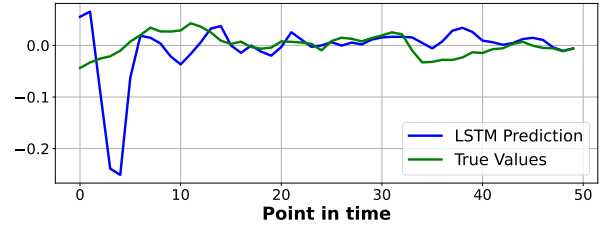


Fig. 6. LSTM performance under the accidental events shown in Fig. 4.

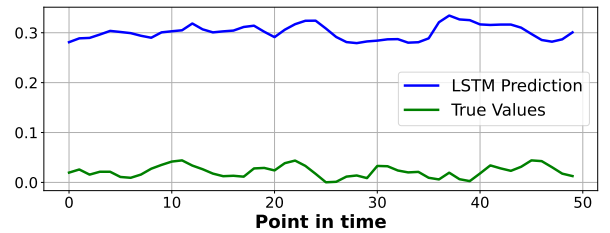


Fig. 7. LSTM performance under FDI attack, shown in Fig. 4.

The threshold value used in classifying accidental events and attacks has been obtained using the data shown in Fig. 5. In this experiment, the length ( $L$ ) of the predicted sequence is 50. We used the predicted values (blue curve) and observed values (green curve) as  $P$  and  $O$  for each of the 100 PMUs in the system to compute the RMSE value in Eq. (1). Fig. 8 depicts the distribution of the threshold values gathered for all 100 PMUs, which follows a Gaussian distribution  $\mathcal{N}(\mu, \sigma^2)$  with  $\mu = 12.15$  and  $\sigma = 7.34$  mv. The RMSE values are then used to pinpoint the deviations from normal cases.

Fig. 9 gauges the efficiency in distinguishing accidental events and FDI attacks using the OCSVM module of DiToF.

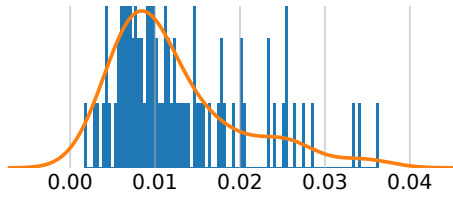


Fig. 8. Distribution of the threshold values used for anomaly detection and classification of attacks & accidental events.

Note that the same input data used for training the LSTM is employed to determine the OCSVM's attack threshold. We conducted this experiment for all components as shown in the  $x$ -axis of Fig. 9. To differentiate between attacks and events, first, the Pearson correlation between the flagged PMU by the LSTM model and all the remaining 99 PMUs is found using the data subset that was used in the inference of LSTM (i.e., points 250 : 350 in case of an event). For example, the correlation between the flagged component  $C_0$ , flagged by LSTM-based anomaly detection, and each of the other components is calculated and fed into the OCSVM module. Then, the number of points labeled as +1 are counted. This experiment is repeated for all 100 PMUs and the output distribution is shown in blue dots in Fig. 9. The experiment is repeated using the false data (FDI attack) and the final distribution is shown in red dots. Note that Fig. 9 shows the results obtained during the inference phase. We observed that a threshold (AT value) of 89 proves to be the optimal value for effectively classifying the outputs into two categories: accidental events and attacks. Such AT is shown in a green dashed line in Fig. 9. Overall, the classification accuracy of the LSTM module is 90%, distinguishing normal from accidental/attack events, while the OCSVM module only misclassifies 3% of its corresponding correlation inputs. Considering the simplicity of both the LSTM and the OCSVM models, our results confirm the effectiveness of DiToF.

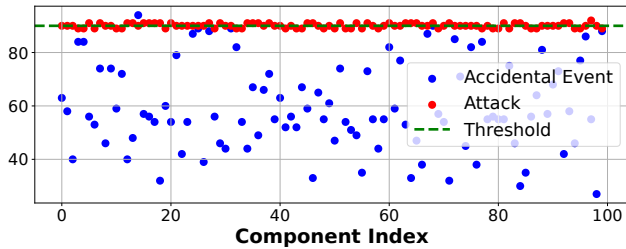


Fig. 9. The classification accuracy achieved by the OCSVM module of DiToF.

## VI. CONCLUSION

Many industrial applications are adopting the CPS design methodology, where a decentralized management strategy is pursued to meet the scalability and complexity requirements. Digital twins have emerged as an effective technology to facilitate the distributed decision-making process in CPS. However, DTs are prone to cyberattacks where the injection of false data may result in inappropriate decisions and could lead to catastrophic consequences. This paper has presented, DiToF, a novel approach for detecting FDI attacks in CPS. DiToF pursues a two-step approach; first, it incorporates recurrent neural networks to detect data anomalies and then uses a data-driven fingerprinting scheme to classify the detected anomalies

as malicious attacks or accidental events. DiToF is generic in terms of being applicable to different CPS applications. The validation results using a smart grid dataset have confirmed the effectiveness of DiToF, where 97% accuracy could be achieved in distinguishing FDI attacks from accidental events. DiToF tackles the case when only one CPS component is compromised. In the future, we will extend the scope to tackle collusive attack scenarios, where multiple actors coordinate to launch FDI attacks.

## REFERENCES

- [1] L. Ribeiro and M. Björkman, "Transitioning from standard automation solutions to cyber-physical production systems: An assessment of critical conceptual and technical challenges," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3816–3827, 2018.
- [2] H. Lin et al., "Challenges and opportunities in the detection of safety-critical cyberphysical attacks," *Comp.*, vol. 53, no. 3, pp. 26–37, 2020.
- [3] S. Khan et al., "Digital twin for advanced automation of future smart grid," in *ICAISC*, 2023, pp. 1–6.
- [4] Q. He et al., "Management and real-time monitoring of interconnected energy hubs using digital twin: Machine learning based approach," *Solar Energy*, vol. 250, pp. 173–181, 2023.
- [5] L. You and M. Zhu, "Digital twin simulation for deep learning framework for predicting solar energy market load in trade-by-trade data," *Solar Energy*, vol. 250, pp. 388–397, 2023.
- [6] Z. Shen et al., "Digital twin application for attack detection and mitigation of pv-based smart systems using fast and accurate hybrid machine learning algorithm," *Solar Energy*, vol. 250, pp. 377–387, 2023.
- [7] M. Ebrahimabadi et al., "Digital twin integrity protection in distributed control systems," in *CCNC*, 2024.
- [8] K. Manandhar, X. Cao, F. Hu, and Y. Liu, "Detection of faults and attacks including false data injection attack in smart grid using kalman filter," *IEEE Trans. Control of Net Sys.*, vol. 1, no. 4, pp. 370–379, 2014.
- [9] P. Hu et al., "Detection of false data injection attacks in smart grids based on expectation maximization," *Sensors*, vol. 23, no. 3, p. 1683, 2023.
- [10] R. Deng and H. Liang, "False data injection attacks with limited susceptance information and new countermeasures in smart grid," *IEEE Trans. on Industrial Informatics*, vol. 15, no. 3, pp. 1619–1628, 2018.
- [11] M. Jorjani, H. Seifi, and A. Y. Varjani, "A graph theory-based approach to detect false data injection attacks in power system ac state estimation," *IEEE Trans. on Industrial Info.*, vol. 17, no. 4, pp. 2465–2475, 2020.
- [12] M. Jin, J. Lavaci, and K. H. Johansson, "Power grid ac-based state estimation: Vulnerability analysis against cyber attacks," *IEEE Trans. s on Automatic Control*, vol. 64, no. 5, pp. 1784–1799, 2018.
- [13] X. Liu and Z. Li, "False data attacks against ac state estimation with incomplete network information," *IEEE Trans. on smart grid*, vol. 8, no. 5, pp. 2239–2248, 2016.
- [14] J. Zhao, L. Mili, and M. Wang, "A generalized false data injection attacks against power system nonlinear state estimator and countermeasures," *IEEE Trans. on Power Systems*, vol. 33, no. 5, pp. 4868–4877, 2018.
- [15] S. Wang, S. Bi, and Y.-J. A. Zhang, "Locational detection of the false data injection attack in a smart grid: A multilabel classification approach," *IEEE IoT Journal*, vol. 7, no. 9, pp. 8218–8227, 2020.
- [16] Y. Li et al., "Detection of false data injection attacks in smart grid: A secure federated deep learning approach," *IEEE Trans. on Smart Grid*, vol. 13, no. 6, pp. 4862–4872, 2022.
- [17] J. James, Y. Hou, and V. O. Li, "Online false data injection attack detection with wavelet transform and deep neural networks," *IEEE Trans. s on Industrial Informatics*, vol. 14, no. 7, pp. 3271–3280, 2018.
- [18] H. Shi et al., "Detection of false data injection attacks in smart grid based on a new dimensionality-reduction method," *Computers & Electrical Engineering*, vol. 91, p. 107058, 2021.
- [19] M. Eckhart et al., "Digital twins for cyber-physical systems security: State of the art and outlook," *Security and Quality in Cyber-Physical Systems Engineering*, pp. 383–412, 2019.
- [20] N. Malsa, V. Vyas, and J. Gautam, "Rmse calculation of lstm models for predicting prices of different cryptocurrencies," *International Journal of System Assurance Engineering and Management*, pp. 1–9, 2021.
- [21] B. Foggo, K. Yamashita, and N. Yu, "pmubage: The benchmarking assortment of generated pmu data for power system events - part i: Overview and results," *ArXiv*, vol. abs/2204.01095, 2022.
- [22] J. J. Q. Yu, Y. Hou, and V. O. K. Li, "Online false data injection attack detection with wavelet transform and deep neural networks," *IEEE Trans. on Industrial Informatics*, vol. 14, no. 7, pp. 3271–3280, 2018.