applied sciences

MDPI

Article

A Resilient Cyber-Physical Demand Forecasting System for Critical Infrastructures against Stealthy False Data Injection Attacks

Iffat Gheyas [1,*], Gregory Epiphaniou [1,*], Carsten Maple [1,*] and Subhash Lakshminarayana [2]

[1] WMG, University of Warwick, Coventry CV4 7AL, UK
[2] School of Engineering, University of Warwick, Coventry CV4 7AL, UK
* Correspondence: iffat.gheyas@warwick.ac.uk (I.G.); gregory.epiphaniou@warwick.ac.uk (G.E.); cm@warwick.ac.uk (C.M.)

Abstract: The safe and efficient function of critical national infrastructure (CNI) relies on the accurate demand forecast. Cyber-physical system-based demand forecasting systems (CDFS), typically found in CNI (such as energy, water, and transport), are highly vulnerable to being compromised under false data injection attacks (FDIAs). The problem is that the majority of existing CDFS employ anomaly-based intrusion detection systems (AIDS) to combat FDIAs. Since the distribution of demand time series keeps changing naturally with time, AIDS treat a major change in the distribution as an attack, but this approach is not effective against colluding FDIAs. To overcome this problem, we propose a novel resilient CDFS called PRDFS (Proposed Resilient Demand Forecasting System). The primary novelty of PRDFS is that it uses signature-based intrusion detection systems (SIDS) that automatically generate attack signatures through the game-theoretic approach for the early detection of malicious nodes. We simulate the performance of PRDFS under colluding FDIA on High Performance Computing (HPC). The simulation results show that the demand forecasting resilience of PRDFS never goes below 80%, regardless of the percentage of malicious nodes. In contrast, the resilience of the benchmark system decreases sharply from about 99% to less than 30%, over the simulation period as the percentage of malicious nodes increases.

Keywords: resilient demand forecasting system; false data injection; intrusion detection systems; data trustworthiness; node reputation

Citation: Gheyas, I.; Epiphaniou, G.; Maple, C.; Lakshminarayana, S. A Resilient Cyber-Physical Demand Forecasting System for Critical Infrastructures against Stealthy False Data Injection Attacks. *Appl. Sci.* **2022**, *12*, 10093. https://doi.org/10.3390/app121910093

Academic Editor: Yangquan Chen

Received: 26 August 2022
Accepted: 29 September 2022
Published: 7 October 2022

1. Introduction

Our critical national infrastructure (CNI) systems like those driving power generation and distribution, water treatment, electricity production, and other platforms are being increasingly digitized and controlled by the Internet of Things (IoT) [1]. The task of forecasting demand for CNI is fundamental and crucially important to run CNI smoothly and make sound operational decisions [2]. Demand specifies the number of goods and services that will be used or consumed in an economy at a given time. Cyber physical system (CPS)-based demand forecasting systems (CDFS) are ubiquitous in CNI [2] where sensor nodes are deployed in the environment to monitor the current demand for the services of CNI (e.g., traffic and electricity consumption) [3–7]. The sensors transmit their measurements at a regular interval to a base station through one or more intermediate nodes depending on the distance between the sensor and the base station. In the base station, the measurements of all sensors are aggregated using aggregate operators like average and sum. The forecasting model stored in the base station uses the values of the aggregate demand time series to develop an estimate of expected demand during a specified future period. The demand forecast is transmitted to the application layer to adjust the supply of the services of CNI such as electricity, water, gas, transportation, etc., to the forecasted demand. Inaccurate demand forecasts can create an imbalance of demand and supply

which may have a devastating impact in terms of lives lost, infrastructure damage, and economic disruption [8–10]. The accuracy of demand forecasting depends on the reliability of the inputs to the forecasting model. In CPS, sensor nodes are deployed in remote areas without any physical protection [11]. Furthermore, sensors are resource-constrained tiny nodes, due to which the implementation of traditional cryptographic techniques is not possible [11]. Hence, sensor nodes are highly vulnerable to attacks, and attackers can easily tamper with the sensor node and compromise it [6,12]. Once a sensor node is compromised, attackers can launch any form of attack they desire [13]. A false data injection attack (FDIA) is the worst possible attack that could be made on a forecasting system. In FDIAs, attackers modify the measurements of the compromised sensors to change the demand forecasts to their desired level [14,15]. To resolve this issue, it is necessary to improve the resilience of CDFS to stealthy attacks. Resilience is twofold. Firstly, the CDFS must be robust against attacks (that is, be able to either prevent the attacks or detect them at a very early stage), and secondly, it should bounce back quickly to its baseline before the attack [4].

Recently, several resilient CDFS have been proposed that consider any drastic change in demand distribution as an attack [9,16–18]. The problem with this approach is that the joint probability distribution of demand time series keeps changing naturally with time [2]. The approach often produces an extremely high false positive rate [9]. In addition, the attackers have figured out how to avoid detection. They launch collusion attacks in which the attacker takes control of multiple sensor nodes and alters the readings of those sensors slightly, such that the aggregate value of all sensors' readings is skewed towards the attacker's desired direction [19]. This aggregate measurement estimated at the time step t is incorporated into the training set. The demand forecasting models are adapted to recent history to make out-of-sample forecasts so that the machine learning forecasting system dynamically adapts to data distributions that evolve over time. Since the forecasting model is evolving toward false data, over time, the normal sensors are marked as compromised. They have an increasingly reduced influence on the model's evolution. This way, through careful selection of reported values, the attackers manage to shift the accepted value far away from the actual value. To make the attack even stealthier, at each time point, attackers randomly select a subset of sensors from the set of compromised sensors, and only the selected compromised nodes inject false data. In contrast, the other compromised sensors provide unadjusted, untampered measurements [20,21]. To protect the CDFS from stealthy attacks, we propose a novel CDFS called PRDFS (Proposed Resilient Demand Forecasting System). The key advantage of the PRDFS over the existing CDFS is its signature-based intrusion detection system (SIDS) trained on autonomously generated attack signatures.

The rest of this paper is organised as follows: Section 2 provides definitions of several important concepts used within this paper, an overview of challenges in demand forecasting, and how the existing resilient CDFS deal with these challenges. Section 2 also contains an overview of existing defence schemes against stealthy FDIA. Section 3 presents our proposed CDRFS (PRDFS). Section 4 explains the research methodology, including a description of the benchmark CDFS, adversary model, performance criteria, and evaluation methodologies for resilient CDFS. Section 5 presents the results, wherein we discuss the advantages and disadvantages of PRDFS and the benchmark CDFS in detail. Section 6 concludes this work.

2. Related Work

Resilient demand forecasting is a relatively new area of research [9,16–18]. Most of the existing resilient CDFS consist of two subsystems—the demand forecasting system (DFS) and the intrusion detection system (IDS). The first subsystem consists of normal time series forecasting algorithms that predict future demand for a particular product/service. Kalman Filtering models and Long Short-Term Memory (LSTM) networks (a special variant of recurrent neural networks) are by far the most popular choices for forecasting demand among the existing resilient CDFS [9,17,22,23]. The Kalman Filter is a model-based linear estimator unsuitable for complex demand signals with unknown characteristics.

The purpose of the second subsystem (IDS) of the resilient CDFS is to mitigate the adverse effects of attacks on demand forecasting. We organise this section as follows. Important concepts are defined in Section 2.1. The major challenges in demand forecasting and how the existing resilient CDFS address these challenges are discussed in Section 2.2. An overview of existing attack detection and mitigation algorithms used in IDS is discussed in Section 2.3.

2.1. Definitions

This section aims to provide an operational definition of "an attack on CDFS" and define three core concepts (trust, trustworthiness, and reputation) that are frequently used in this paper.

An attack on a system is defined as "a threat action that undesirably alters system operation by adversely modifying system functions or data" [24]. For this study, an attack on CDFS is defined as systematically deviating the point forecasts and interval forecasts for k-step ahead aggregate demands from the actual values to a certain direction (upwards or downwards) over time by adversely modifying sensor data. The success of the attack is measured on the following two indicators: (1) the higher the deviations of the single-point demand forecast from the actual demand for a given period, the more successful the attack, and (2) the ultimate success of the attack is achieved when the actual demand falls outside of the forecasted bounds.

Although the terms 'trust', 'trustworthiness', and 'reputation' are widely used in the literature, there remains debate about the definitions of all three [11,13,22,23,25–29]. Different studies have different definitions of these terms. In this study, we adopt the following definitions for trust, trustworthiness, and reputation. We use 'trust' and 'trustworthiness' interchangeably. If a node needs to evaluate the trust score of another node, the evaluating node is called the subject node, and the evaluated node is called the object node [20]. 'Trust' or 'trustworthiness' of a node is the subjective probability assigned to an object node by a subject node at time step t based on various factors (e.g., the object node's long-term reputation, the similarity of its measurements with the measurements generated by other nodes received at the same time step etc.) that the measurement received from the object node is accurate [27,30]. The trustworthiness of data provenance is the probability assigned by a subject node to the provenance that the measurement received from the data provenance is accurate. Data provenance refers to the metadata that captures the complete history of a piece of data from origin to destination. The data provenance of measurement consists of two basic components [31]. The first component is the route of the measurement. A route refers to the path a measurement travels on a network. The route includes every node that handles the measurement between its source and destination. The second component of the data provenance is the list of actions performed by participating nodes. The trustworthiness of a measurement value is the probability that the measurement is accurate [11]. The reputation of an object node is the overall probability estimated from the trust scores assigned to the object node by all subject nodes over time [26,30].

2.2. Challenges in Demand Forecasting

Demand forecasting is a mature area of research [32]. Nonetheless, for accurate predictions of demand, many challenges remain to be addressed [32–34]. One of the major challenges is the curse of dimensionality. In demand forecasting, the predictor variables are lagged variables. The lagged variables are highly correlated. Consequently, even a moderate number of predictor variables may trigger the curse of dimensionality, and demand forecasting models often suffer from overfitting. Most of the existing CDFS do not take special measures to protect themselves against the curse [9,17]. They train a single network (e.g., LSTM) with a set of lagged aggregate demand time series values to forecast the out-of-sample data.

Another major challenge in demand forecasting is the presence of non-stationarity in demand time series [32–34]. In a non-stationary time series, the frequency content changes

in time; all frequencies do not always exist. There are currently three main approaches for dealing with time-varying frequency components [35]. The first approach is to apply the differencing operator on-demand time series and remove these patterns from the time series. However, the main difficulty of the differencing approach is that it requires exact knowledge about the locations of all frequency components in the time series, which is by no means easy to capture.

The second approach to deal with the time-varying frequency components is to enter a large number of non-seasonal lagged variables in a neural network (e.g., LSTM) during the training so that it automatically selects an optimal number of seasonal and non-seasonal lagged variables and models time-varying frequency components. Most of the existing resilient CDFS adopt this mechanism [9,17]. This approach leads to a large number of lagged input variables in the model which might exacerbate the curse of dimensionality.

The third approach to deal with the time-varying frequency components is to use a hybrid of wavelet transform and neural networks [35,36]. This approach has been drawing increasingly more attention in recent years. This approach uses wavelet techniques to split the original time series into multiple component time series. A neural network (e.g., LSTM) is trained with the non-seasonal lagged variables generated from the component time series to forecast the future values of the original demand curve. Currently, more comparative studies are required to confirm the superiority of one approach over the other.

Perhaps the biggest challenge in developing a demand forecasting model is that the older values of the demand time series are less useful than the newer ones as training data and become obsolete quickly, since the distribution of demand time series keeps changing with time [30]. Several studies suggest ensuring that the forecasting model adjusts rapidly to the changing distribution. Based on these studies, the higher penalty should be assigned to in-sample forecast errors on more recent demand observations during the training phase. However, this approach poses many practical problems. The key problem is how to weigh the values of demand time series appropriately in the presence of time-varying frequency components. An inadequate weighting scheme may cause more harm than good, as it may destroy the distribution of the original time series. Hence, the common practice is to keep adapting the forecasting model to the new observations as the sensors gather them without applying an arbitrary weighting scheme [9].

2.3. Overview of Existing Defense Strategies against Stealthy False Data Injection Attacks (FDIAs)

Existing defense systems to deal with stealthy false data injection attacks (FDIAs) can be grouped into two broad categories: anomaly-based intrusion detection systems (AIDS) and provenance-based trust management systems (TMS) [9,13,22,31,37,38].

The goal of AIDS is to detect malicious nodes and exclude their readings from the aggregation of sensor measurements while estimating the most trustworthy aggregate measurement at a given time step. When AIDS is used to deal with FDIA, the simple average is usually used to aggregate the measurements of the normal sensors. Existing AIDS can be broadly split into two categories: confidence interval-based AIDS [9,22,37] and clustering-based AIDS [38]. Both types of AIDS are based on the hard clustering concept that classifies a sensor node as either malicious (1) or normal (0). Confidence interval-based AIDS divides the search space into only two clusters (one for each class: malicious and normal). On the contrary, clustering-based AIDS allow multiple clusters for each class; this type of AIDS performs better on search space with local patterns.

AIDS uses unsupervised hard clustering algorithms to perform clustering of the search space to classify the sensor nodes as malicious and normal. Different search spaces have been explored for detecting stealthy malicious nodes, including measurement space, forecasting residual space, Spatio-temporal correlation space, the skewness space of residuals in the forecasting demand model, and reputation space [9,22,37,38]. The major limitation of AIDS is that they use a set of arbitrary rules to cluster the search space and to label the clusters as malicious and normal. Past studies suggest that AIDS suffer from low malicious node detection accuracy and a high false positive rate (FPR) [9].

The goal of TMS, in contrast to AIDS, is not to classify the sensor nodes as malicious and normal [13,31]. Instead, they carry out unsupervised fuzzy clustering in the sensor measurement space to assign a reputation score to each sensor to control its influence on the sensor measurement aggregation process. During the aggregation process, TMS assigns a trustworthiness score to each measurement at time step t based on the data similarity and provenance similarity among measurements, and the trustworthiness of its data provenance, which is usually set to the reputation score of the least reputed node in the provenance. TMS then determines the most trustworthy measurement at time step t using the weighted average method. Most of the existing TMS use linear models to update the reputation scores of nodes based on the trustworthiness scores of their measurements received at time step t. However, the linear models may not be able to accurately estimate the reputation scores of nodes in the face of stealthy FDIA.

3. Proposed Resilient Demand Forecasting Systems

We name our proposed resilient demand forecasting system PRDFS (Proposed Resilient Demand Forecasting System). PRDFS consists of five modules: (1) Trust Management System (TMS), (2) Demand Forecasting System (DFS), (3) Data Generator (DG), (4) Ethical Hacking System (EHS), and (5) Signature-based Intrusion Detection System (SIDS). Codes and pseudocodes of PRDFS are provided in Supplementary Materials. Figure 1 shows the architecture of PRDFS. The modules are linked to each other via the input and output they exchange. Figure 1 depicts the exchanges of inputs and outputs among the modules. The internal workings of the modules are shown in the flowcharts of Figures 2–6. Table 1 shows the definitions of the terms used in this section.

The first layer of PRDFS is only for graphical representation (Figure 1). Each node of this layer represents a data provenance (DPV). In our simulation study, for simplicity purposes, each DPV contains only one sensor node. The DPVs transmit streams of measurements of demand $\left(\overrightarrow{MV(t)}\right)$ to the following two units: TMS and DG.

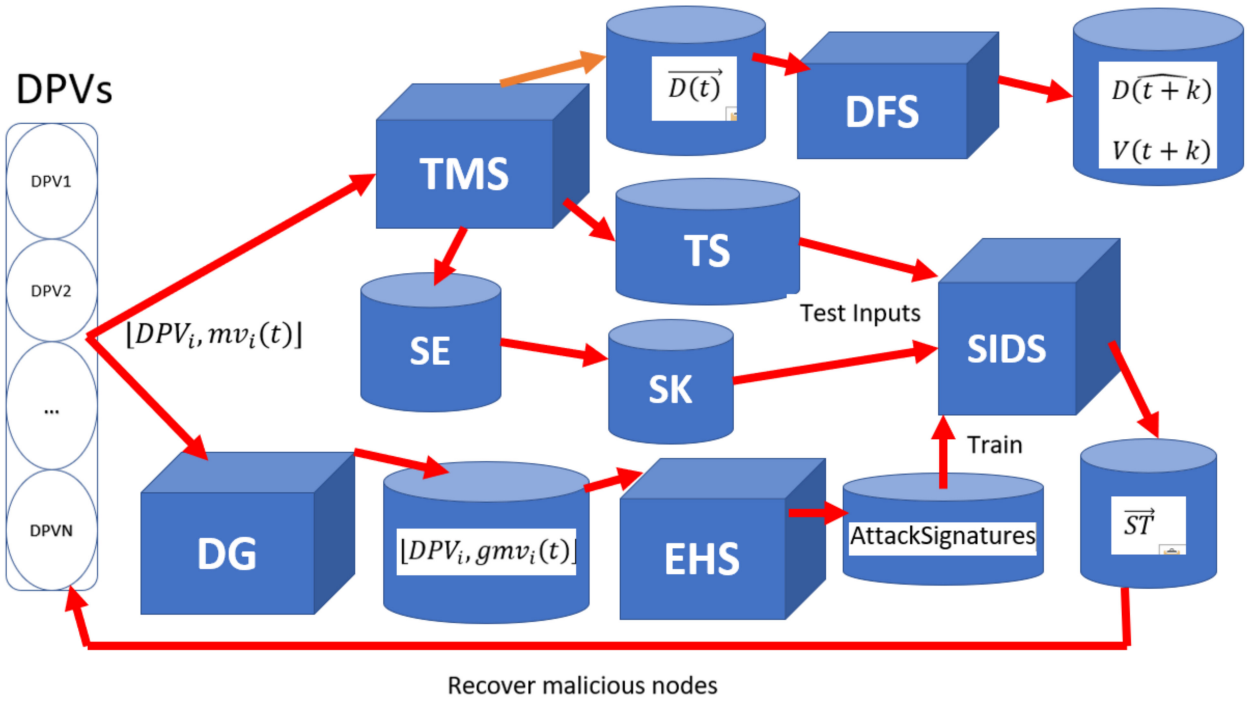

Figure 1. Architecture of PRDFS.

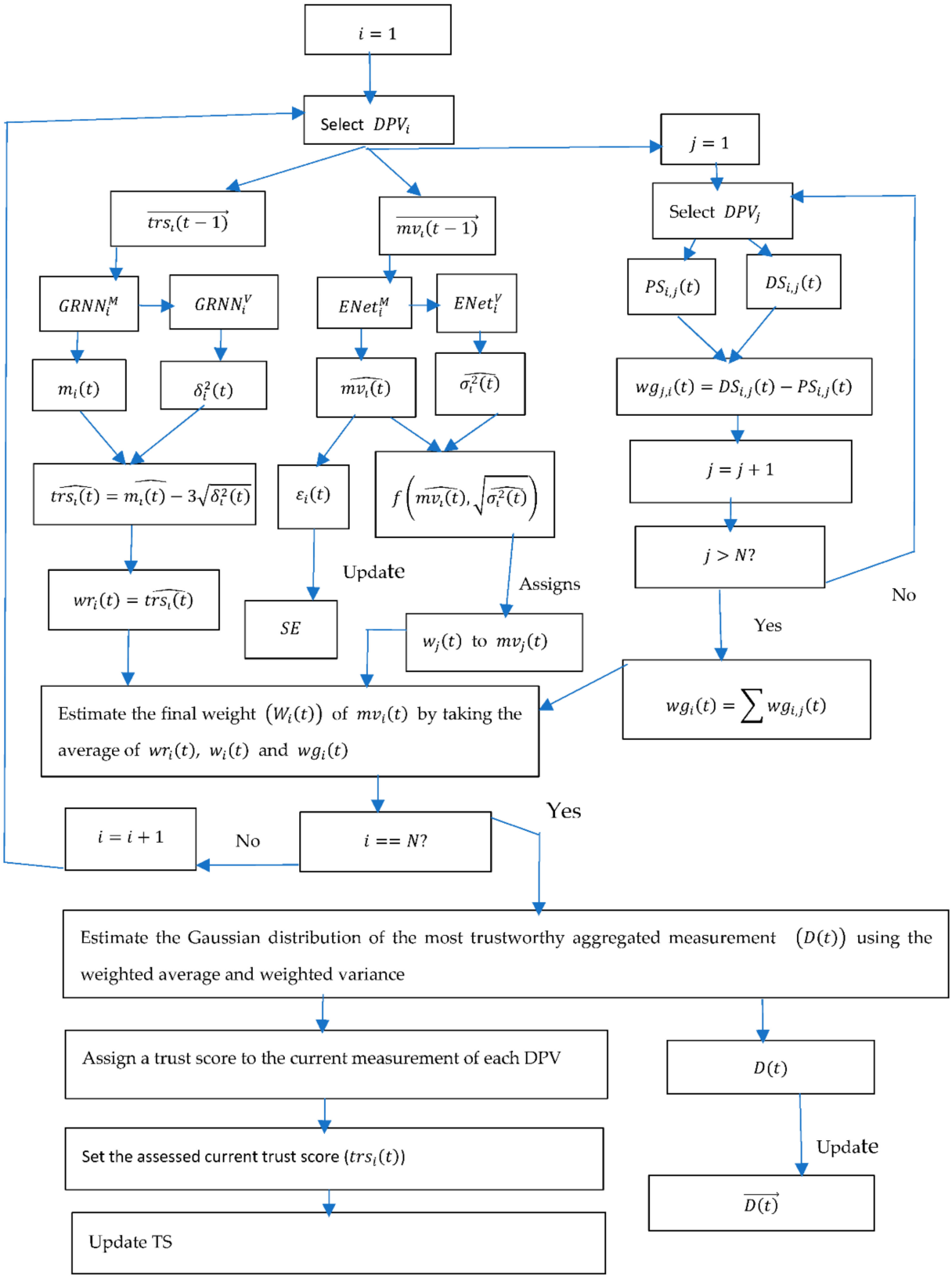

Figure 2. Flowchart of TMS.

TMS: The flowchart for TMS is presented in Figure 2. TMS produces the following three outputs, which the other modules use as inputs.

Table 1. The Definitions of Terms used in Section 3.

Term	Definition
N	Total number of DPVs
t	Current time step
DPV_i	Data provenance of the ith measurement
DPV_j	Data provenance of the jth measurement
nc	The total number of nodes common to DPV_i and DPV_j
n_1	The lenght of the longer DPV between DPV_i and DPV_j
$\overrightarrow{mv_i(t)}$	Time series of the ith DPV's measurements containing both past and current measurements
$\overrightarrow{mv_i(t-1)}$	Time series of the ith DPV's past measurements
$mv_i(t)$	The received measurement of the ith DPV at time step t
$mv_j(t)$	The received measurement of the jth DPV at time step t
$DS_{i,j}(t)$	The data similarity between $mv_i(t)$ and $mv_j(t)$ estimated using MAAPE
$PS_{i,j}(t)$	Provenance similarity between the DPVs of $mv_j(t)$ and $mv_i(t)$: $PS_{i,j}(t) = nc/n_1$
$\widehat{mv_i(t)}$	One step ahead forecast of the conditional mean of $mv_i(t)$
$\varepsilon_i(t)$	$\varepsilon_i(t) = mv_i(t) - \widehat{mv_i(t)}$
$\overrightarrow{\varepsilon_i(t)}$	The residual series for the one step ahead forecasts of ith DPVs measurements containing both the past and current residuals
$\overrightarrow{\varepsilon_i(t-1)}$	The residual series for the one step ahead forecasts of ith DPVs measurements containing the past residuals
$\sigma_i^2(t)$	The observed conditional variance of $mv_i(t)$: $\sigma_i^2(t) = \varepsilon_i^2(t)$
$\widehat{\sigma_i^2(t)}$	One step ahead forecast of $\sigma_i^2(t)$
$ENet_i^M$	The ensemble of CNNs forecasting $\widehat{mv_i(t)}$
$ENet_i^V$	The ensemble of CNNs forecasting $\widehat{\sigma_i^2(t)}$ based on $\overrightarrow{\varepsilon_i^2(t-1)}$
$\epsilon_i(t)$	$\epsilon_i(t) = \sigma_i^2(t) - \widehat{\sigma_i^2(t)}$
$f\left(\widehat{mv_i(t)}, \sqrt{\widehat{\sigma_i^2(t)}}\right)$	A gaussian distribution with mean $\widehat{mv_i(t)}$ and standard deviation $\sqrt{\widehat{\sigma_i^2(t)}}$
SE	A matrix containing the residual series for the one step ahead forecasts of all DPVs measurements
SK	A matrix containing the time series of skewness of the series in SE
$\overrightarrow{MV(t)}$	The measurements of all DPVs at t : $\overrightarrow{MV}(t) = [mv_1(t), \ldots, mv_N(t)]$
$D(t)$	The most trustworthy aggregate demand measurement at time step t based on $\overrightarrow{MV(t)}$
$\overrightarrow{D(t)}$	The time series of aggregate demand
$\overrightarrow{trs_i(t)}$	Time series of the assessed trust scores of the ith DPV including its assessed trust score at time step t
$\overrightarrow{trs_i(t-1)}$	Time series of the assessed trust scores of the ith DPV, not including its assessed trust score at time step t
$trs_i(t)$	The assessed trust score of the ith DPV at time step t
$\widehat{trs_i(t)}$	The forecasted trust score of the ith sensor at time step t
$m_i(t)$	The forecasted conditional mean of $trs_i(t)$ based on $\overrightarrow{trs_i(t-1)}$
$\delta_i^2(t)$	The forecasted conditional variance of $m_i(t)$
$GRNN_i^M$	The ith member of GRNN of the ensemble of GRNNs forecasting $m_i(t)$
$GRNN_i^V$	The ith member of GRNN of the ensemble of GRNNs forecasting $\delta_i^2(t)$

Table 1. *Cont.*

Term	Definition
TS	$TS = \left[\overrightarrow{trs_1(t)}, \ldots, \overrightarrow{trs_N(t)}\right]$
$D(t)$	The most trustworthy aggregate demand measurement at time step t
$\overrightarrow{D(t)}$	The time series of aggregate demand (based on $\overrightarrow{MV(t)}$) including $D(t)$
$\overline{D(t+k)}$	The k-step-ahead forecasted conditional mean aggregate demand based on $\overrightarrow{D(t)}$
$V(t+k)$	The k-step-ahead forecasted value of the conditional variance of $\overline{D(t+k)}$
$L(t+k)$	$L(t+k) = \overline{D(t+k)} - 3\sqrt{V(t+k)}$
$U(t+k)$	$U(t+k) = \overline{D(t+k)} + 3\sqrt{V(t+k)}$
$Ensemble^{mean}$	The ensemble of CNNs forecasting $\overline{D(t+k)}$
$Ensemble^{var}$	The ensemble of CNNs forecasting $V(t+k)$
$gmv_i(t)$	Measurement generated for the ith DPV at time step t by DG
$\overrightarrow{GMV(t)}$	$\overrightarrow{GMV(t)}$ contains the measurements generated by DG for all sensors at t: $\overrightarrow{GMV(t)} = [gmv_1(t), \ldots, gmv_N(t)]$
$\overline{D_g(t)}$	The most trustworthy aggregate demand measurement at time step t based on $\overrightarrow{GMV(t)}$
$\overrightarrow{D_g(t)}$	The time series of aggregate demand (based on $\overrightarrow{GMV(t)}$) including $\overline{D_g(t)}$
$\widehat{D_g(t+k)}$	The k-step-ahead forecasted conditional mean aggregate demand based on $\overrightarrow{D_g(t)}$
$\overrightarrow{pos_q(t,I)}$	The position of the qth particle at time step t and iteration I.
$\overline{D_q(t)}$	The most trustworthy aggregate demand measurement at time step t based on $\overrightarrow{pos_q(t,I)}$
$\overrightarrow{D_q(t)}$	The time series of aggregate demand (based on $\overrightarrow{pos_q(t,I)}$) including $\overline{D_q(t)}$
$\widehat{D_q(t+k)}$	The k-step-ahead forecasted conditional mean aggregate demand based on
$\overrightarrow{Vel_q(t,I)}$	$\overrightarrow{Vel_q(t,I)}$ is the velocity of the qth particle.
$fitness(\overrightarrow{pos_q(t,\ I)}$	The fitness score of $\overrightarrow{pos_q(t,\ I)}$
$\overrightarrow{pbest_q}$	The personal best position of the qth particle
$fitness\left(\overrightarrow{pbest_q}\right)$	The fitness score of $\overrightarrow{pbest_q}$
$\overrightarrow{gbest}$	The global best position of all the particles
$fitness\left(\overrightarrow{gbest}\right)$	The fitness score of $\overrightarrow{gbest}$
ST	ST is a matrix containing the time series of the states (malicious and normal) of each sensor at previous time steps
$\overrightarrow{ST_q(t,I)}$	A vector containing the state of each DPV in the qth particle at time step t and iteration I
$\overrightarrow{ST(t=1,\ I=1)}$	A vector containing the state of each DPV at the beginning of a game
$\overrightarrow{ST}$	A vector containing the state of each DPV in $\overrightarrow{gbest}$

Output 1: TMS derives the aggregate demand time series $\left(\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}\right)$ of individual DPVs by averaging the measurements. DFS uses $\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}$ as inputs. If all DPVs were equally trustworthy, a simple average would be appropriate to aggregate the measurements of all DPVs. However, since the measurements of all DPVs are not always equally trustworthy due to adversarial attacks and mechanical issues, a weighted average is more appropriate than a simple average for the derivation of aggregate demand ($\boldsymbol{D}(\boldsymbol{t})$). TMS first determines the trustworthiness weights of the measurements based on the data similarity, provenance sim-

ilarity, past measurements, and the predicted trustworthiness of the corresponding DPVs at a given time step using the ensembles of Convolutional Neural Networks (CNNs) [39] and Generalized Regression Neural Networks (GRNNs) [40], and then performs the weighted average. TMS sends the aggregate demand time series $\left(\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}\right)$ to DFS.

Output 2: At the end of each time step, TMS assesses the trust scores $[\boldsymbol{trs_1}(\boldsymbol{t}),\ldots,\boldsymbol{trs_N}(\boldsymbol{t})]$ for all DPVs based on the distances between their measurements and the most trustworthy aggregated measurement $(\boldsymbol{D}(\boldsymbol{t}))$ at that time step. TMS updates $\boldsymbol{TS} = \left[\overrightarrow{\boldsymbol{trs_1}(\boldsymbol{t-1})},\ldots,\overrightarrow{\boldsymbol{trs_N}(\boldsymbol{t-1})}\right]$ by adding $[\boldsymbol{trs_1}(\boldsymbol{t}),\ldots,\boldsymbol{trs_N}(\boldsymbol{t})]$ to $\boldsymbol{TS}$. The updated $\boldsymbol{TS}$ is $\boldsymbol{TS} = \left[\overrightarrow{\boldsymbol{trs_1}(\boldsymbol{t})},\ldots,\overrightarrow{\boldsymbol{trs_N}(\boldsymbol{t})}\right]$.

Output 3: At each time step, TMS produces the residuals $[\boldsymbol{\varepsilon_1}(\boldsymbol{t}),\ldots,\boldsymbol{\varepsilon_N}(\boldsymbol{t})]$ of one-step ahead forecasts of the measurements of all DPVs at that time step. TMS then updates $\boldsymbol{SE} = \left[\overrightarrow{\boldsymbol{\varepsilon_1}(\boldsymbol{t-1})},\ldots,\overrightarrow{\boldsymbol{\varepsilon_N}(\boldsymbol{t-1})}\right]$ to $\boldsymbol{SE} = \left[\overrightarrow{\boldsymbol{\varepsilon_1}(\boldsymbol{t})},\ldots,\overrightarrow{\boldsymbol{\varepsilon_N}(\boldsymbol{t})}\right]$ by adding $[\boldsymbol{\varepsilon_1}(\boldsymbol{t}),\ldots,\boldsymbol{\varepsilon_N}(\boldsymbol{t})]$ to $\boldsymbol{SE}$.

DFS: The flowchart for DFS is provided in Figure 3. DFS performs $\boldsymbol{k}$-step ahead out-of-sample forecasting of demand based on $\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}$ derived by TMS. It uses wavelet decomposition [41] to split $\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}$ into component series of various frequencies. It then uses a pair of hierarchical ensembles of CNNs to estimate the distribution of demand forecasts based on the values of the component time series at non-seasonal lags. The member CNNs are trained sequentially such that the first member CNN is trained on the component time series of $\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}$ and each of the other member CNNs is trained on the component time series of the residuals of the member CNN trained just before it. DFS keeps increasing the number of member CNNs in the ensemble until the residual series of the last member CNN becomes white noise. We call this ensemble of CNNs '$\boldsymbol{Ensemble^{mean}}$'. $\boldsymbol{Ensemble^{mean}}$ forecasts the conditional mean demand $\left(\overline{\boldsymbol{D}(\boldsymbol{t+k})}\right)$. DFS also trains another ensemble of CNNs called '$\boldsymbol{Ensemble^{var}}$' in the same way it trained $\boldsymbol{Ensemble^{mean}}$. The only difference is that the first member CNN of $\boldsymbol{Ensemble^{var}}$ is trained on the component time series of the squared residuals of $\boldsymbol{Ensemble^{mean}}$ and each of the remaining member CNNs is trained on the component time series derived from the time series of the absolute values of residuals obtained from the above member CNN. $\boldsymbol{Ensemble^{var}}$ forecasts the conditional variance $(\boldsymbol{V}(\boldsymbol{t+k}))$ of $\left(\overline{\boldsymbol{D}(\boldsymbol{t+k})}\right)$. During forecasting, each ensemble member in $\boldsymbol{Ensemble^{mean}}$ and $\boldsymbol{Ensemble^{var}}$ forecasts values separately. The overall forecast $\left(\overline{\boldsymbol{D}(\boldsymbol{t+k})}\right)$ of $\boldsymbol{Ensemble^{mean}}$ is the summation of the forecasts of its all members. The overall forecast $(\boldsymbol{V}(\boldsymbol{t+k}))$ of $\boldsymbol{Ensemble^{var}}$ is the summation of the absolute values of the forecasts of all its member CNNs. The point forecast (i.e., the most expected demand) of the DFS module at $\boldsymbol{t+k}$ is $\overline{\boldsymbol{D}(\boldsymbol{t+k})}$. The interval forecast of DFS at $\boldsymbol{t+k}$:

The lower bound of the forecast distribution:

$$\boldsymbol{L}(\boldsymbol{t+k}) = \overline{\boldsymbol{D}(\boldsymbol{t+k})} - \boldsymbol{3}\sqrt{\boldsymbol{V}(\boldsymbol{t+k})} \tag{1}$$

The upper bound of the forecast distribution:

$$\boldsymbol{U}(\boldsymbol{t+k}) = \overline{\boldsymbol{D}(\boldsymbol{t+k})} + \boldsymbol{3}\sqrt{\boldsymbol{V}(\boldsymbol{t+k})} \tag{2}$$

***DG*:** The flowchart for DG is provided in Figure 4. The DG learns the joint probability distribution of the individual time series of each DPV and generates an infinite stream of $\left(\overrightarrow{\boldsymbol{GMV}(\boldsymbol{t})}\right)$ for each DPV. DG sends $\overrightarrow{\boldsymbol{GMV}(\boldsymbol{t})}$ to EHS to facilitate the generation of attack signatures.

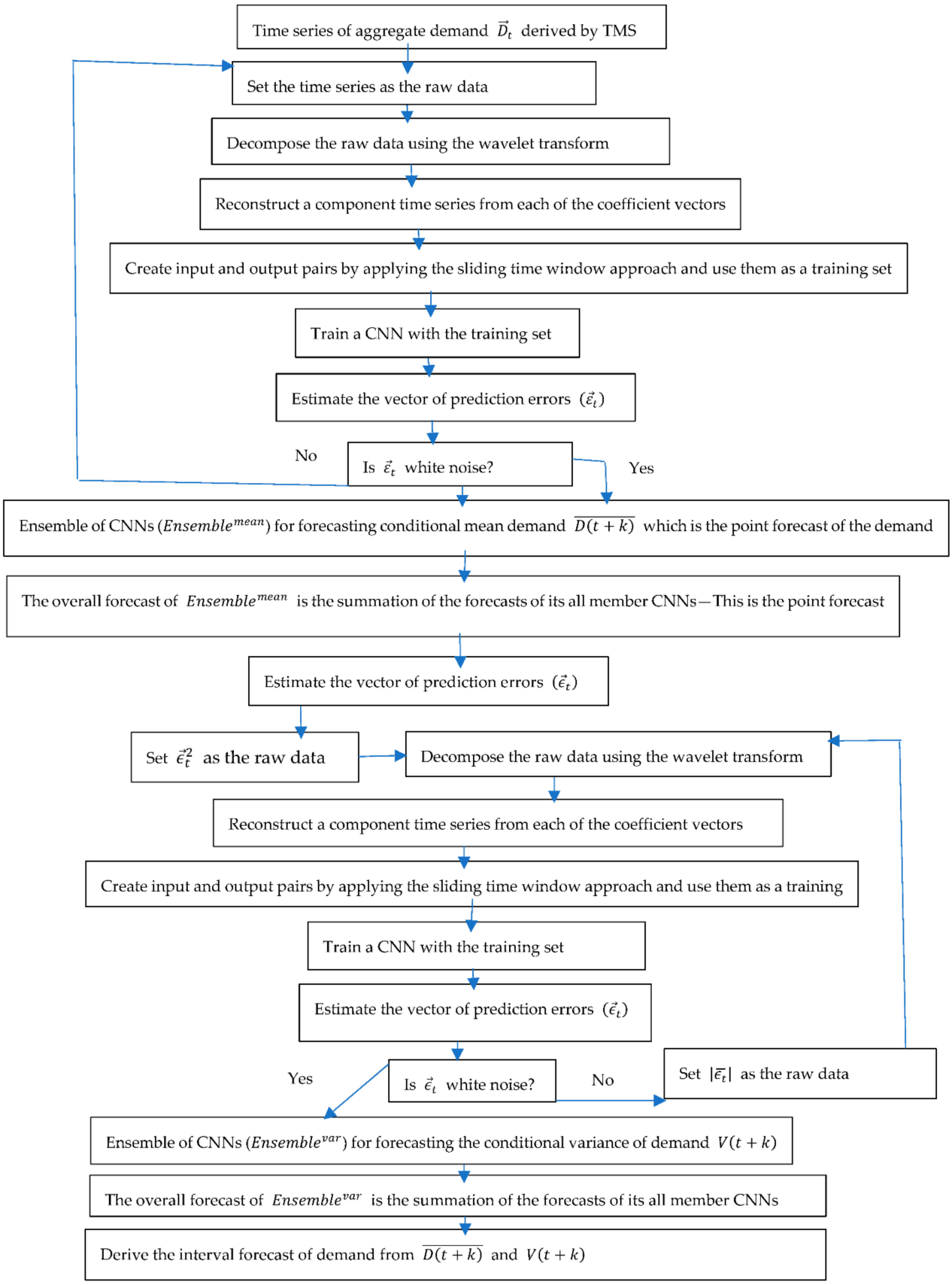

Figure 3. Flowchart for DFS.

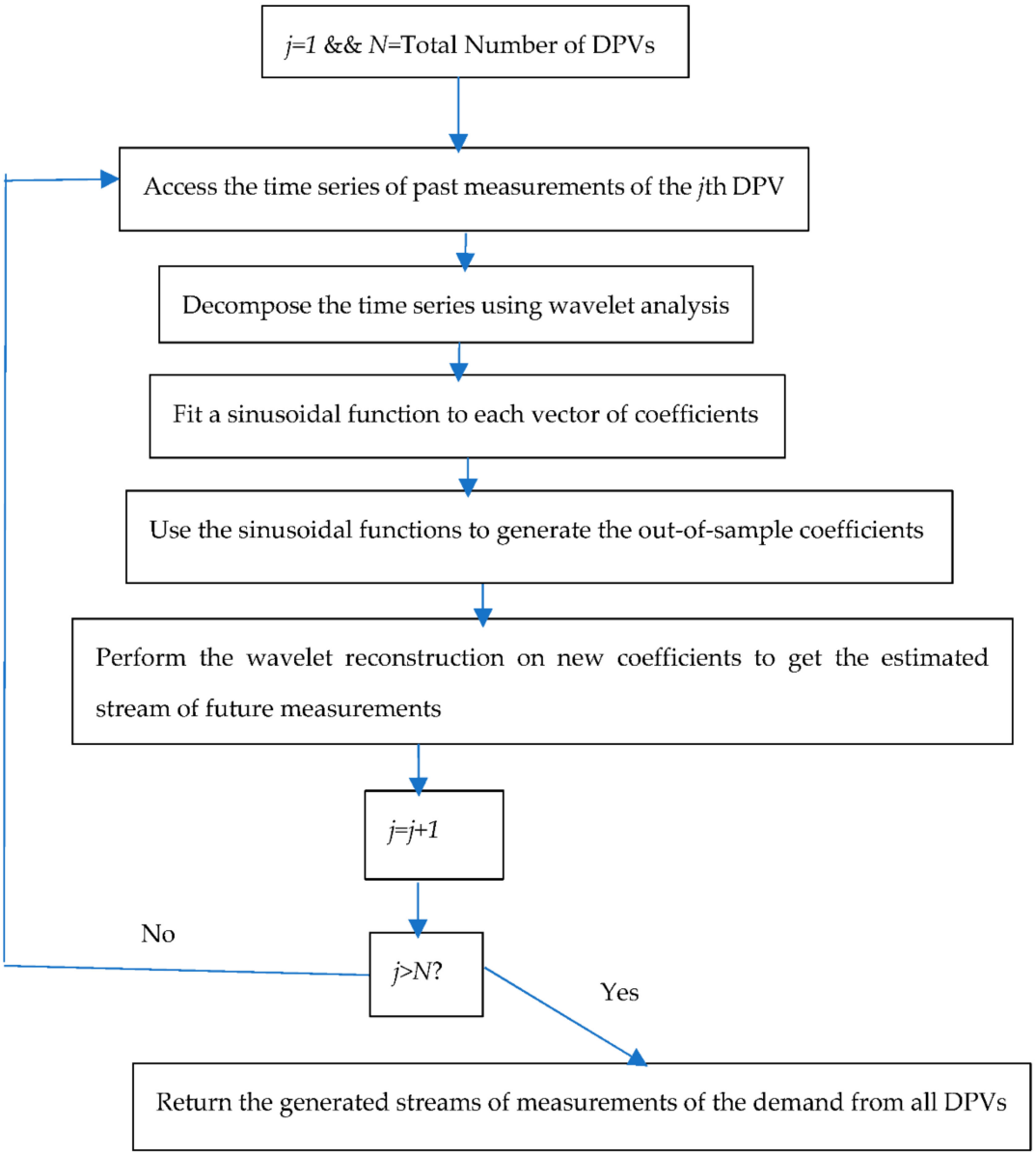

Figure 4. Flowchart for DG.

***EHS*:** The flowchart for EHS is given in Figure 5. The task of EHS is to generate all possible attack signatures for the training of SIDS. To do so, the EHS uses the Particle Swarm Optimization (PSO) algorithm [42] with a game theoretic approach [43]. It plays repeated games with the other modules of PRDFS. In these games, EHS plays as an attacker while TMS and SIDS are teamed up to play as a defender of DFS. The attacker and defender play under the following rules: (1) in each game, the attacker has a fixed number of malicious nodes at the beginning. (2) If the defender detects a malicious node, it becomes a normal node. From the next step onwards, the attacker cannot modify its measurements. (3) An attacker cannot compromise a new or recovered node in the middle of the game. (4) EHS works with complete knowledge of the PRDFS. (5) The attacker randomly chooses one of the following two objectives to pursue in a game: (i) shift the demand trend upward, and (ii) shift the demand trend downward.

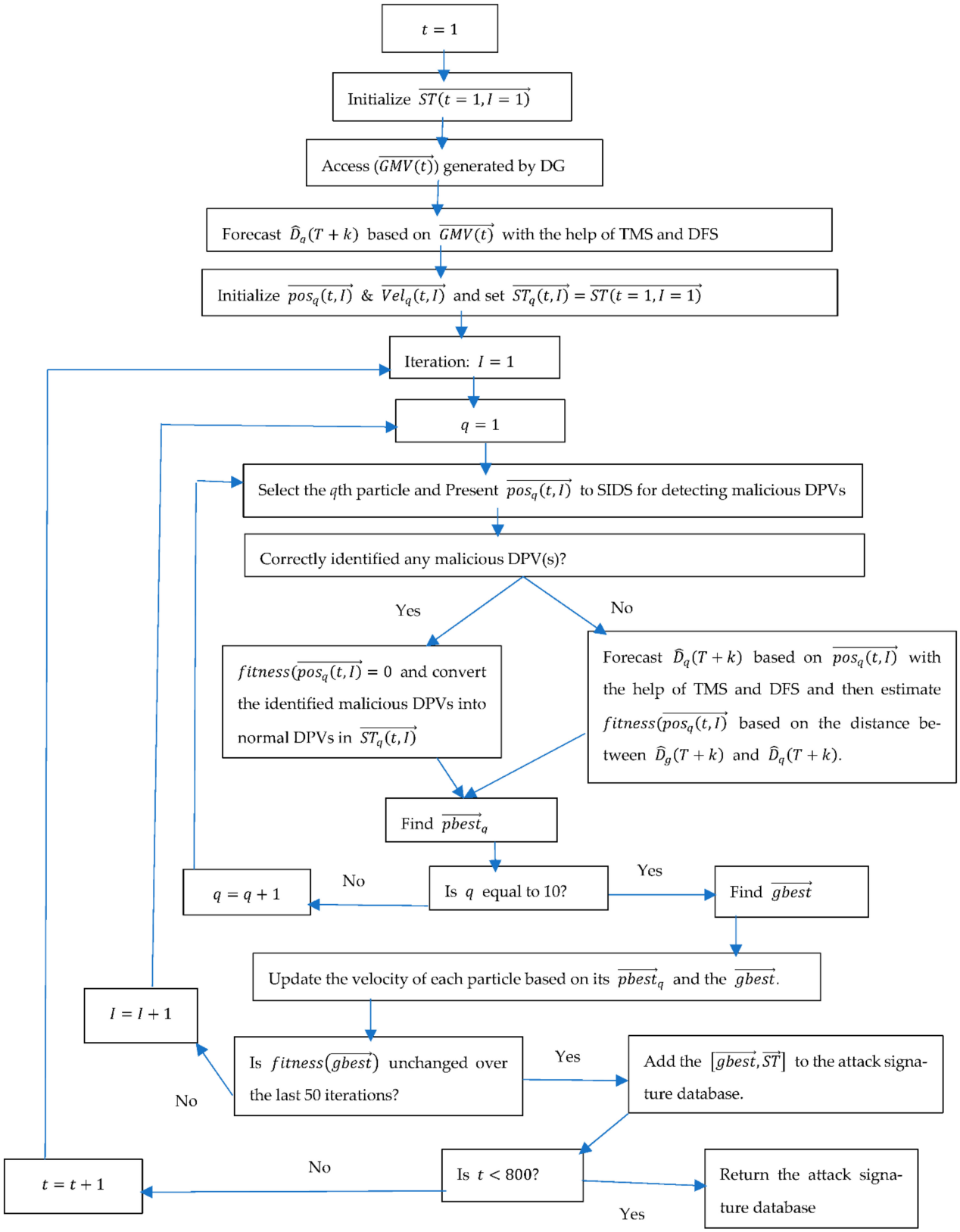

Figure 5. Flowchart for EHS.

To generate optimal attack signatures, the PSO (employed by EHS) follows the following steps.

Step 1. Initialize the positions and velocities of ten particles. The position $\left(\overrightarrow{\boldsymbol{pos_q(t,I)}} = [\boldsymbol{gmv_1(t)} \ldots \boldsymbol{gmv_N(t)}]\right)$ of each particle $(\boldsymbol{q})$ at time step $\boldsymbol{t}$ and iteration $\boldsymbol{I}$ is

a vector containing the measurements of all DPVs at time step t, wherein the measurements of normal DPVs are the same as the generated measurements by the DG and the measurements of the malicious DPVs are randomly initialized. The velocity $\left(\overrightarrow{Vel_q(t,I)} = [Vel_1(t),\ldots,Vel_N(t)]\right)$ of each particle (q) at time step t and iteration I is a vector that has the same length as the position vector. Each element of the velocity vector contains the direction and speed at which the measurement of the corresponding DPV is updated from the one iteration to the next during the PSO search. In each particle, the velocities of normal DPVs are set to zero and regarded as constant. In addition, each particle has a state vector $\left(\overrightarrow{ST_q(t,I)}\right)$ containing the state of each DPV. Initially, the state vectors of all particles are the same.

Step 2. The fitness of each particle's position $\left(fitness\left(\overrightarrow{pos_q(t,I)}\right)\right)$ is determined using the following method:

- Present $\overrightarrow{GMV(t)}$ to TMS. TMS estimates the most trustworthy aggregated measurement of demand $\overline{D_g(t)}$ and sends it to DFS to forecast the k-step ahead demand $\widehat{D_g(t+k)}$.
- Present $\overrightarrow{pos_q(t,I)}$ to TMS. TMS estimates the most trustworthy aggregated measurement of demand $\overline{D_q(t)}$ and sends it to DFS to forecast the k-step ahead demand $\widehat{D_q(t+k)}$. TMS also updates the time series of assessed trust scores (TS) of DPVs by adding the assessed trust scores of DPVs (found in $\overrightarrow{pos_q(t,I)}$) at time step t to TS, as well as the time series of the residuals of one-step ahead forecasts of each DPV's measurements (SE) by adding the residuals of forecasting the measurements of DPVs at time step t to SE. SIDS uses the updated TS and SE to predict the current states (malicious and normal) of DPVs.
- If SIDS detects any malicious node correctly, set $fitness\left(\overrightarrow{pos_q(t,I)}\right)$ to zero and change the state(s) of the detected malicious node(s) from malicious to normal in $\left(\overrightarrow{ST_q(t,I)}\right)$. Else if $\widehat{D_q(t+k)} < \widehat{D_g(t+k)}$ when the objective is shifting the demand upward or $\widehat{D_q(t+k)} > \widehat{D_g(t+k)}$ when the objective is shifting the demand downward, set $fitness\left(\overrightarrow{pos_q(t,I)}\right)$ to zero. Else estimate the fitness using the following formula:

$$fitness\left(\overrightarrow{pos_q(t,I)}\right) = 100 \times \tan^{-1}\left(\left|\frac{\widehat{D_g(t)} - \widehat{D_q(t)}}{\widehat{D_g(t)}}\right|\right) \quad (3)$$

Step 3. Find the personal best position of each particle at the current Ith iteration from the positions it has visited before and set the personal best position $\left(\overrightarrow{pbest_q}\right)$ of the qth particle to it. If it is the first iteration, set the $\overrightarrow{pbest_q}$ to its current position. We denote the fitness of $\overrightarrow{pbest_q}$ by $\left(fitness\left(\overrightarrow{pbest_q}\right)\right)$. Find the global best position of all particles $\left(\overrightarrow{gbest}\right)$ from the positions they have visited before. We denote the fitness of $\overrightarrow{gbest}$ by $\left(fitness\left(\overrightarrow{gbest}\right)\right)$.

Step 4. Update the velocity of each malicious DPV in qth particle using the following equation:

$$\overrightarrow{Vel_q(t,I+1)} = \overrightarrow{Vel_q(t,I)} + C_1 \times rand(1,1) \times \left|\overrightarrow{pbest_q} - \overrightarrow{pos_q(t,I)}\right| + C_2 \times rand(1,1) \times \left|\overrightarrow{gbest} - \overrightarrow{pos_q(t,I)}\right| \quad (4)$$

where $rand(1,1)$ is a random number between (0,1). C_1 and C_2 are learning factors. We set the learning factor to 2 ($C_1 = C_2 = 2$).

Step 5. Update each particle's position using the following equation:

$$\overrightarrow{pos_q(t,I+1)} = \overrightarrow{pos_q(t,I)} + \overrightarrow{Vel_q(t,I+1)} \quad (5)$$

Repeat steps 2–5 until the stopping criterion is satisfied.

Step 6. $\overrightarrow{\boldsymbol{gbest}}$ contains the optimal measurements for the malicious sensors. $\left[\overrightarrow{\boldsymbol{gbest}},\overrightarrow{\boldsymbol{ST}}\right]$ is an optimal attack signature, wherein $\overrightarrow{\boldsymbol{ST}}$ is a vector containing the state of each DPV in $\overrightarrow{\boldsymbol{gbest}}$.

EHS generates attack signatures through repeated games and keeps updating the attack signature database by adding new signatures.

SIDS: The flowchart for SIDS is provided in Figure 6. The task of SIDS is to classify DPVs into one of the two groups: malicious (1) and normal (0). It uses an ensemble of GRNNs to perform the classification task. Each member GRNN is trained to classify one DPV as either normal or malicious. Target outputs vary between member GRNNs. During training, each GRNN's target outputs are the states (malicious state or normal state) of the corresponding DPV at a given time step. However, the input vectors of all member GRNNs are the same. During training, the raw input is $\overrightarrow{\boldsymbol{gbest}}$. During testing, the raw input is the vector of measurements transmitted by the DPVs. SIDS sends the raw input to TMS. TMS updates ***TS*** and ***SE*** accordingly. SIDS estimates the time series of the skewness coefficients of residuals at each time step by moving a sliding window over the residual series in ***SE***. This yields the matrix ***SK*** containing the skewness series of each DPV. The input vector contains 100 consecutive most recent lagged variables from ***TS*** and ***SK***.

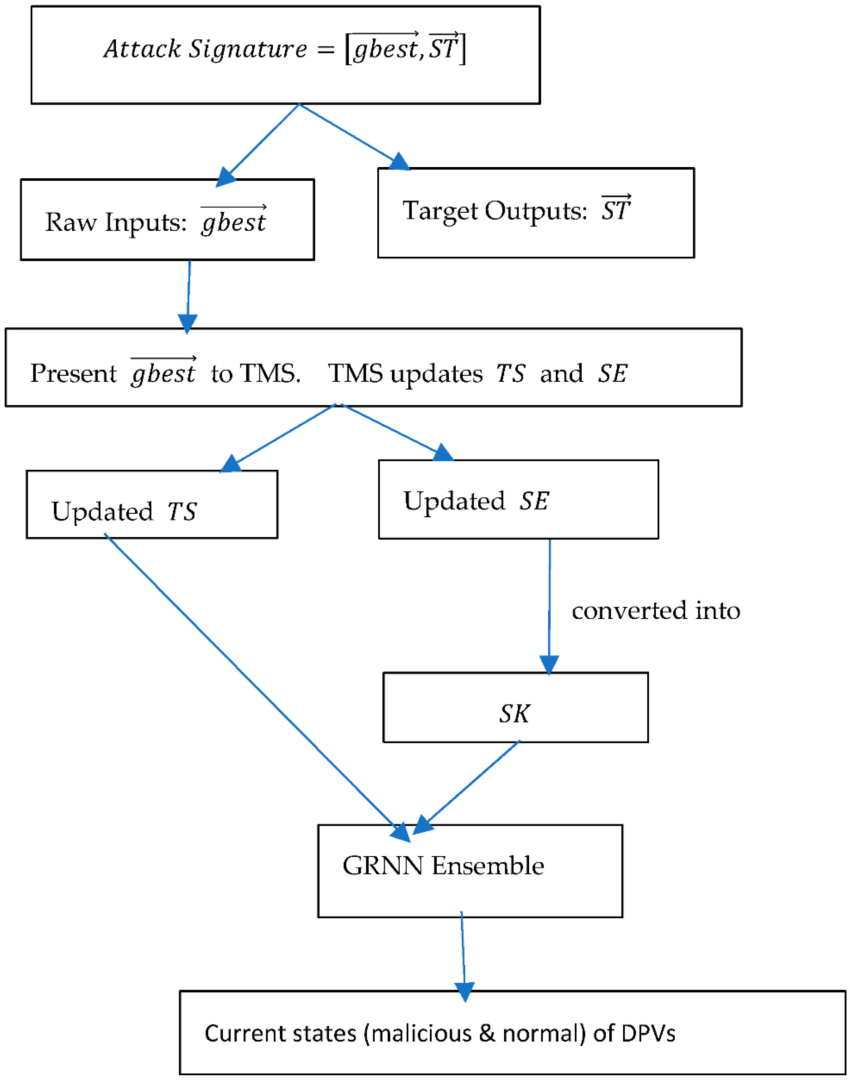

Figure 6. Flowchart for SIDS.

It is worth noting that when malicious DPVs are detected, they get converted to normal DPVs at the very time step, so that from the next time on, they provide the actual measurements. However, there are no ways to recover the actual measurements of the recovered DPVs at time step t when they are first detected as having compromised. Hence, SIDS cannot improve the performance of estimating the most trustworthy aggregate demand $(\boldsymbol{D}(\boldsymbol{t}))$ at time step t. By detecting and recovering the malicious DPVs at time step t, SIDS ensures the better performance of TMS in estimating the most trustworthy aggregate demand at the later time steps. In PRDFS, TMS does not exclude the measurements obtained from the malicious DPVs while estimating $\boldsymbol{D}(\boldsymbol{t})$. This is because there is a chance of the detected malicious DPVs being in benign modes at time step t, that means they have reported the actual measurements. Even if the detected malicious DPVs are in malicious modes at time step t, the distance between the malicious measurements may not be far away from the actual measurements if the attack is at the initial stage. Hence, to avoid losing information from malicious DPVs by discarding their readings, TMS adjusts the weights of the measurements to control their influence on the estimation of $\boldsymbol{D}(\boldsymbol{t})$. Thus, by providing a trustworthy time series of historical aggregate demands $\left(\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}\right)$, TMS ensures the performance of DFS in forecasting the $\boldsymbol{k}$-step ahead values of $\left(\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}\right)$ is least affected by the attack.

4. Research Methodology

We compare PRDFS with a benchmark resilient demand forecasting system. No existing resilient CDFS have all the features that have been demonstrated to be effective for resilient demand forecasting systems. Hence, we implement a benchmark resilient CDFS by combining the promising solutions proposed in previous studies that complement each other and strengthen the performance of the resilient demand forecasting system. We call this benchmark algorithm BRDFS. The remaining section is organized as follows: the evaluation methodology for PRDFS and BRDFS is discussed in Section 4.1. The performance metrics against which the performance of PRDFS and BRDFS is measured are discussed in Section 4.2. A brief description of BRDFS is presented in Section 4.3. PRDFS and BRDFS are evaluated on synthetic data in the presence of on-off colluding FDIA. The details of the adversary model that generated stealthy FDIA attacks are presented in Section 4.4. The synthetic data generation process is described in Section 4.5.

4.1. Evaluation Methodology for PRDFS and BRDFS

To compare PRDFS and BRDFS, we simulate two CPS. For simplicity purposes, each CPS consists of two layers of nodes. The first layer contains 14 sensors that monitor demand continuously. The second layer contains a base station. All sensor nodes are directly connected to the base station. They transmit their measurements to the base station at regular intervals. PRDFS resides in the base station of one CPS, and BRDFS resides in the base station of the other CPS. PRDFS and BRDFS are implemented in MATLAB and their performance is tested on High Performance Computing (HPC). PRDFS and BRDFS both play multiple games with the adversary model under different percentages of malicious nodes. The important rules of the games are stated below: (1) at the beginning of each game, the attacker compromises a fixed number of malicious nodes. (2) If the defender detects a malicious node, it becomes a normal node. From the next step onwards, the attacker cannot modify its measurements. (3) An attacker cannot compromise a new or recovered sensor node. (4) The duration of each game is 1000-time steps. (5) The adversary model has partial knowledge. At the start of the game, the adversary model gets the most recent 500 measurements of all malicious sensors, whereas PRDF and BRDFS get the most recent 1000 measurements of all sensors. (6) At each time step, the data generator (described in Section 4.5) generates the measurements of sensors at that time step. The adversary model modifies the measurements of malicious sensors. All measurements are then transmitted to the base station. PRDFS and BRDFS estimate the most trustworthy measurement based

on the noisy sensor measurements, append the most trustworthy measurement to the aggregate demand time series, retrain the forecasting models on the updated aggregate demand time series, and forecast one-step ahead of demand using the updated forecasting model, and classify each sensor node into one of the two types: malicious and normal. The adversary model also adapts itself based on the new measurements of malicious sensors.

4.2. Performance Measures

PRDFS and BRDFS are compared in terms of two measures. The first performance measure is the accuracy of detecting malicious nodes. The second performance measure is the resilience of the demand forecasting system in the face of ongoing attacks. These measures are described in detail in Sections 4.2.1 and 4.2.2.

4.2.1. Accuracy in Detecting Malicious Nodes

PRDFS and BRDFS are compared with respect to their ability to correctly classify sensors as malicious (i.e., involved in stealthy FDIA attacks) or normal (i.e., not involved in attacks). When the classification is compared with the true status, there are four possible outcomes: true positive (TP), true negative (TN), false positive (FP), and false negative (FN) (details are available in Table 2).

Table 2. Performance Metric.

		Predicted Label	
		Positive (Malicious)	**Negative (Normal)**
Actual Status	Positive (Malicious)	True Positive (TP)	False Negative (FN)
	Negative (Normal)	False Positive (FP)	True Negative (TN)

We use accuracy rate and false positive rate (FPR) as the performance criteria metric based on the above performance metric shown in Table 2. The accuracy rate and FPR are measured using the following formulae.

$$Accuracy\ rate = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$FPR = \frac{FP}{FP + tN} \tag{7}$$

4.2.2. The Resilience of the Demand Forecasting System in the Face of Ongoing Attack

The resilience of PRDFS and BRDFS are measured using the following metric for resilience [44]:

$$R = P_n/P \tag{8}$$

where P_n and P present the out-of-sample one step ahead demand forecasting accuracy with and without attacks, respectively. The predictive accuracy is measured in both point estimates and confidence intervals. In the single point estimate, the predicted output is the most expected value of demand. We use the mean arctangent absolute percentage error (MAAPE) to measure the accuracy of single point forecasts [45]. In the MAAPE approach, the accuracy is defined as follows:

$$Accuracy = 100 - MAAPE \times 100 \tag{9}$$

$$MAAPE = \frac{1}{N}\sum_{t=1}^{N}(AAPE_t) \qquad for\ t = 1,\ldots,N \tag{10}$$

$$AAPE_t = \tan^{-1}\left(\left|\frac{A_t - F_t}{A_t}\right|\right) \tag{11}$$

where A_t and F_t denote the actual and forecasted values at data point t, respectively. N is the number of data points.

In the interval estimate, the predicted outputs are the upper and lower bounds of forecasts. In this approach, if the actual demand lies between the forecasted upper and lower bounds, the forecast is correct. Interval estimates are made for the 99.7% confidence level. The 99.7% limits lie three standard deviations below and three above the mean.

4.3. Description of the Benchmark Algorithm (BRDFS)

BRDFS consists of three modules: a demand forecasting system (DFS), a trust management system (TMS), and a malicious node detection system. The details of these systems are given below.

4.3.1. Demand Forecasting System (DFS) of BRDFS

Most of the existing resilient demand forecasting systems use a single Long-Term Short-Term Memory (LSTM) for forecasting the point estimate of the demand [9,17]. For comparison purposes, the DFS of BRDFS consists of two LSTMs for forecasting the interval estimate of the demand. LSTM1 is trained to forecast the conditional mean of demand and LSTM2 is trained to forecast the conditional variance of demand. LSTM1 is trained on the original aggregate demand time series and LSTM2 is trained on the squared residuals of LSTM1. Most of the existing resilient demand forecasting systems do not perform differencing or wavelet decomposition on the demand time series to deal with non-stationarity. Hence, BRDFS does not apply these measures as well.

4.3.2. Trust Management System (TMS) of BRDFS

BRDFS uses a popular TMS proposed in [31] to estimate the most trustworthy measurement at time step t based on the measurements from all available sensors. This TMS is implemented as follows. Normalize all measurements at time step t using the min-max normalization. Find the Euclidean distance between each pair of measurements. Estimate the similarity $(DS_{i,j})$ between the measurements $(mv_i(t)$ *and* $mv_j(t))$ of sensors i and j using the following equation:

$$DS_{i,j} = 1 - \textit{Euclidean distance between } mv_i(t) \textit{ and } mv_j(t) \tag{12}$$

Estimate the provenance similarity $(PS_{i,j})$ between $mv_i(t)$ and $mv_j(t)$ which is the ratio of the total number of common nodes in both provenances to the total number of nodes in the provenance with the longer length. Adjust $DS_{i,j}$ to $PS_{i,j}$ using the following pseudocode:

$$\textit{if } DS_{i,j} \geq 0.6 \textit{ then } DS_{i,j} = DS_{i,j} + (1 - PS_{i,j})$$

$$\textit{else } DS_{i,j} = DS_{i,j} - PS_{i,j} \textit{ end}$$

Since in our simulation, we assume that all sensor nodes are directly linked to the based station, $PS_{i,j}$ is always zero. Find the average similarity $\overline{DS}_j$ of each measurement $(mv_j(t))$ to all other measurements.

$$\overline{DS}_j = \frac{\sum_{i=1}^{n-1} \widehat{DS}_{i,j}}{n-1} \tag{13}$$

where n represents the total number of measurements.

Find the measurement $mv_j(t)$ with the highest $\overline{DS}_j$ as the mean represents the total number of measurements (M) measurement (i.e., the most trustworthy measurement). Estimate the standard deviation (σ) of M. Generate a Gaussian function $f(M,\sigma)$ with M and σ. Assign an intermediate trust score $\hat{s}_d(t)$ to each measurement at time step t

using $f(M,\sigma)$. Estimate the final score $\bar{s}_d(t)$ of each measurement using the following linear model.

$$\bar{s}_d(t) = c_d \bar{s}_n(t-1) + (1-c_d)\hat{s}_d(t) \quad (14)$$

where $\bar{s}_n(t-1)$ is current reputation score of the sensor node made the measurement. c_d is a constant to be specified by the user. We set c_d to 0.5.

Set $\bar{s}_d(t)$ as the intermediate reputation score $\left(\hat{s}_n(t)\right)$ of the corresponding sensor. Estimate the final reputation score $\left(\bar{S}_n(t)\right)$ of the sensor node at the end of time step t using the following equation and we set the value of c_d to 0.5.

$$\bar{s}_n(t) = c_d \bar{s}_n(t-1) + (1-c_d)\hat{s}_n(t) \quad (15)$$

4.3.3. Malicious Node Detection System of BRDFS

BRDFS uses a skewness detector proposed in [37] and a correlation detector CAD proposed in [38] to detect the compromised nodes. If a node is found compromised by the skewness detector and/or CAD, the node is considered as compromised. The details of the above anomaly detectors are given below.

Skewness Detector

This skewness detector is proposed in [37]. The idea is, if a sensor is involved in a collusion attack, the distribution of forecasting residuals of the measurements from the sensor (that usually tend to be normal) should be skewed to the left or right.

The training phase of the skewness detector includes the following steps. Step 1: Fit a forecasting model to the time series of the sensor's readings. Step 2: Find the residuals of the forecasting model. Step 3: Estimate the rolling skewness coefficients of the residual series using the rolling window of N time steps. We set N to 30. This gives a series of skewness coefficients. Step 4: Estimate the unconditional mean (m) and standard deviation (s) of the skewness series. Then compute the acceptable range $(-\delta = m - 3s, and + \delta = m + 3s)$ for the skewness of the forecasting error distribution.

During the stealthy attack distribution, the following steps are followed. Step 1: Take the new sequence (r_0) of forecasting residuals over the last N time steps. Step 2: Generate a sequence (r_{rand}) of normal random number with μ and σ^2 estimated in the training phase. The length of r_{rand} is $0.05N$. Step 3: Replace a small proportion of entries in r_0 with r_{rand} and generate a new sequence r_{test} for testing. Step 4: Compute the skewness coefficient (SC) of r_{test}. If SC falls outside the acceptable range $(-\delta,+\delta)$, raise the alarm that the sensor is participating in the collusion attack.

Correlation Detector 'CAD'

The correlation detector, called CAD, is proposed in [38] to detect compromised nodes. CAD includes two sub-schemes: temporal collusion detection scheme and spatial collusion detection scheme. If both sub-schemes find a node anomalous, the node is regarded as compromised.

Temporal collusion detection sub-scheme performs the following steps to detect the anomalous nodes. Step 1: Estimate the correlation between each node's measurements at time step t and at time step $t+1$. Step 2: Estimate a predefined confidence interval for temporal correlations based on the bootstrap distribution. If at a given time window, the temporal correlation falls outside the confidence interval, the node is considered a malicious node.

The spatial collusion detection sub-scheme performs the following steps. Step 1: Compute the Pearson's correlation coefficient between the measurements of each pair of sensors over time window t (a time window consists of multiple time steps). This gives a correlation matrix which is called the correlation map. Each vector of the matrix contains the pairwise correlation coefficients between the measurements of one particular sensor (i) and that of every other sensor over time window t—this vector is the location of sensor i in

the location map at time window t. Step 2: Estimate the global centroid which is the vector in the matrix that has the smallest distance from all vectors in the matrix. Step 3: Perform k-means clustering to partition the vectors into k clusters. Step 4: If a vector's distance from the global centroid is greater than 0.5, and if the vector's distance from the centroid of its closest cluster is greater than 0.3, the sensor represented by the vector is classified as compromised.

4.4. Adversary Model

Let there be N total number of sensors in a complete sensor network. The attacker can only compromise k out of N sensors. Step 1, randomly choose k sensors to be compromised: generate a random number for each sensor. Sort the sensors based on the corresponding random numbers in ascending order. Select the first k sensors for compromising. Step 2, record each compromised sensor's untampered readings over the next 500-time steps. Build and train a pair of convolutional neural networks (CNN) for each compromised sensor. The first CNN predicts the conditional mean of the individual time series at time step t—the inputs of this network are the past values of the time series, and the desired outputs are one-step-ahead values of the time series. The second network predicts the conditional variance of the conditional mean at time step t—the inputs of this network are the past squared residuals of the first CNN, and the desired outputs are one-step-ahead squared residuals from the first CNN. Step 3, to estimate the modified readings for the compromised sensors, construct a 95% confidence interval for the reading of each compromised sensor (l) at time step t based on the corresponding conditional mean $\left(\hat{m}_l(t)\right)$ and conditional variance $\left(\hat{v}_l^2(t)\right)$ forecast. Step 4, Generate k random numbers in the range $[0,1]$, one for each compromised sensor. If the random number is less than 0.5, the corresponding compromised sensor (l) uses the following formula to calculate its tampered measurement (mv_l) at time step t:

$$mv_l = \hat{m}_l(t) + \hat{\vartheta}_l(t) \times r \tag{16}$$

where r is the normal random number with mean 0 and standard deviation 1.

Otherwise, it reports the upper or lower bound of the 99.7% confidence interval as the reading of the compromised sensor at time step t to shift the trend upward or downwards, respectively. Step 5, if the defender fails to identify the lth compromised sensor at time step t, retrain the corresponding CNN pair with the reported measurement at time step t. Step 6, repeat steps 3–5 to estimate the readings for the compromised sensors at each time step.

4.5. Data Generator

We build a data generator that produces data for an infinite time, so that we can evaluate the performance of the proposed forecasting system. In non-stationary time series, various local frequencies exist at different resolution levels. We use wavelets to generate synthetic non-stationary time series. The mother wavelet and the resolution levels (n) along with the values of approximation coefficients at the highest level and detail coefficients at each level must be specified to generate a time series. We complete the following steps to generate a time series for each sensor. Step 1: Choose a wavelet and a set of scales (or level of resolution) (say 1 to n) where the most valuable frequencies will be assigned. We use Daubechies mother wavelet of order 2 $(db2)$. The number of resolution levels (n) is chosen to be three. Step 2: We define a separate sinusoidal function for the generation of the approximation coefficients at level 3 and the detail coefficients at each resolution level. The higher the resolution level, the smaller the number of coefficients, due to subsampling. To generate a signal of length 10,000 with 'db2' as mother wavelet and three levels of resolution, we need to specify 1252 level-3 approximation coefficients, 1252 level-3 detail coefficients, 2502 level-2 detail coefficients, and 5001 level-1 detail coefficients.

$$w_{j,t} = A\, sin(B(t + C)) + D + \sigma r \tag{17}$$

where $w_{j,t}$ is the (approximation or detail) coefficient at scale j and time point t. A = *amplitude*; C = *phase shift*; D = *vertical shift*; B = $2\pi f$; *t=time step*; σ = *standard deviation*; r = *Gaussian noise*. The values of function parameters (A,f,C,D,σ) are arbitrarily assigned.

Step 3: Reconstruct the signal based on the selected mother wavelet and resolution level using the MATLAB function 'waverec'. We perform augmented Dickey-Fuller tests on the time series and its first and second order derivatives to check whether the time series is stationary or not. The test confirms that the time series and its derivatives are non-stationary. Step 4: At each time step, estimate the mean (μ) of all measurements of demand obtained from 14 sensors to get the aggregate demand time series.

5. Results and Discussion

The performances of PRDFS and BRDFS were comparatively tested on 11 games with an adversary model. Each game was conducted with different percentages of malicious nodes. The duration of each game was 1000-time steps. The simulation results provide interesting insight into the comparative efficacy of PRDFS and BRDFS.

5.1. The Comparative Performance of PRDFS and BRDFS in Terms of Demand Forecasting Resilience to Stealth Attacks

Major Findings

- PRDFS exhibits an almost perfect mean resilience value of **~0.99** over the time frame of the game (i.e., 1000-time steps) regardless of the percentage of the nodes compromised by the adversary (Figure 7). This is mainly because it detects and recovers all compromised nodes at the very early stages of games (within first 30-time steps) (Figure 10).
- During the first 40-time steps of games while the attack is still occurring or just ended, the mean demand forecasting resilience of PRDFS slightly decreases but it never goes below 0.8 (its resilience ranges from 0.82 to 1), regardless of the percentage of the malicious nodes (Figure 7). In contrast, the mean demand forecasting resilience sharply decreases in BRDFS over the simulation period (i.e., 1000 time-steps) as the percentage of malicious nodes increases (Figure 7).
- In PRDFS, the interval demand forecast accuracy demonstrates greater resilience than the point demand forecast accuracy, while the opposite trend is observed in BRDFS (Figure 7). In BRDFS, the interval demand forecast accuracy has weaker resilience compared to the point demand forecast accuracy and the trend becomes more pronounced as the percentage of malicious nodes increases. This is mainly because BRDFS fails to detect most malicious nodes (Figure 11). All colluding malicious nodes report similar measurements and thereby artificially reduce the uncertainty around the mean estimate, which results in low interval demand forecast accuracy and yields a very high FPR in detecting malicious nodes.

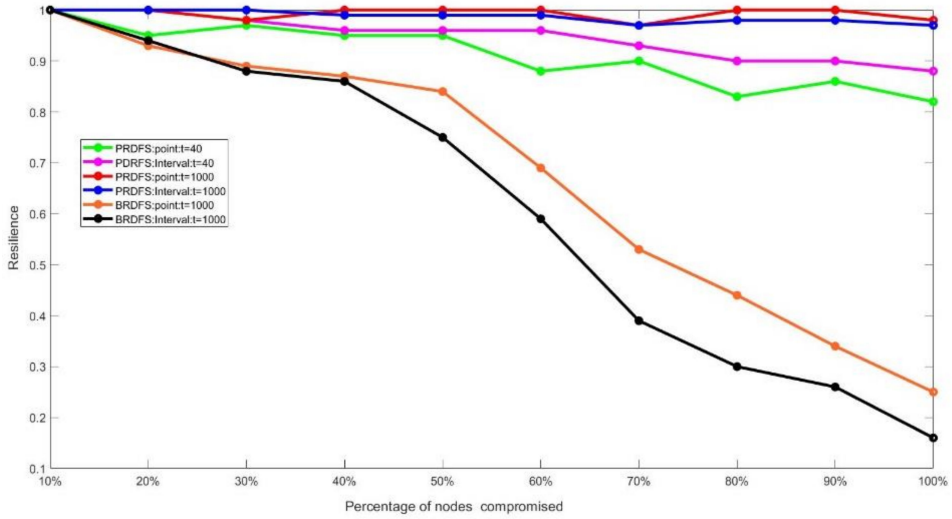

Figure 7. A comparison of resilience between PRDFS and BRDFS under different percentages of malicious nodes. Figure 7 shows the mean resilience of PRDFS and BRDFS in demand forecast accuracy under different percentages of malicious nodes. The X-axis represents the percentages of malicious nodes, and the Y-axis represents the mean resilience. The mean resilience of a system under a given percentage of malicious nodes is the ratio of mean (point or interval) demand forecasting accuracy under attacks with the given percentage of malicious nodes to the mean (point or interval) demand forecasting accuracy under no attack. The red and blue lines plot the mean resilience of PRDFS in respect to point demand forecast accuracy and interval demand forecast accuracy, respectively, over 1000-time steps under different percentages of malicious nodes. The pink and green lines plot the mean resilience of PRDFS in terms of point demand forecast accuracy and interval demand forecast accuracy, respectively, over the first 40-time steps under different percentages of malicious nodes. The orange and black line plots the mean resilience of BRDFS in terms of point demand forecast accuracy and interval demand forecast accuracy, respectively, over 1000-time steps under different percentages of malicious nodes. The figure shows that PRDFS demonstrates near perfect mean resilience over 1000-time steps in both point and interval forecasting accuracy for all percentages of malicious nodes. However, the mean resilience of PRDFS over the first 40-time steps shows the signs of weakening with the increase in malicious nodes. In BRDFS, the mean resilience over 1000-time steps falls sharply as the percentage of malicious nodes increases. PRDFS is observed to have higher resilience in the interval demand forecasting accuracy than in the point forecasting accuracy. The opposite trend is seen in BRDFS.

We attribute the high resilience of PRDFS in the face of an attack to the combined effect of the strong performances from its modules: SIDS, TMS, and DFS. Sections 5.2–5.4 discuss the comparative performances of the individual subsystems.

5.2. The Comparative Performance of PRDFS and BRDFS in Detecting Malicious Nodes

5.2.1. Major Findings:

- PRDFS maintains a high malicious node detection accuracy of around 95% (ranging from 97% to 89%) while still maintaining a low FPR, ranging from 3–9%, with the increasing percentage of malicious nodes (Figures 8 and 9). In contrast, the accuracy of BRDFS sharply falls with the rising FPR as the percentage of malicious nodes increases (Figures 8 and 9).
- PRDFS successfully detects all malicious nodes within the first 30-time steps regardless of the percentage of the malicious nodes, whereas BRDFS fails to detect any malicious node after 15-time steps (Figure 10). It appears that BRDFS has a shorter time window for detecting malicious nodes than PRDFS.

- BRDFS manages to correctly detect malicious nodes when the percentage of malicious nodes is low (Figures 10 and 11). BRDFS fails to detect any malicious node when the percentage of malicious nodes is 30% or over (Figure 11).

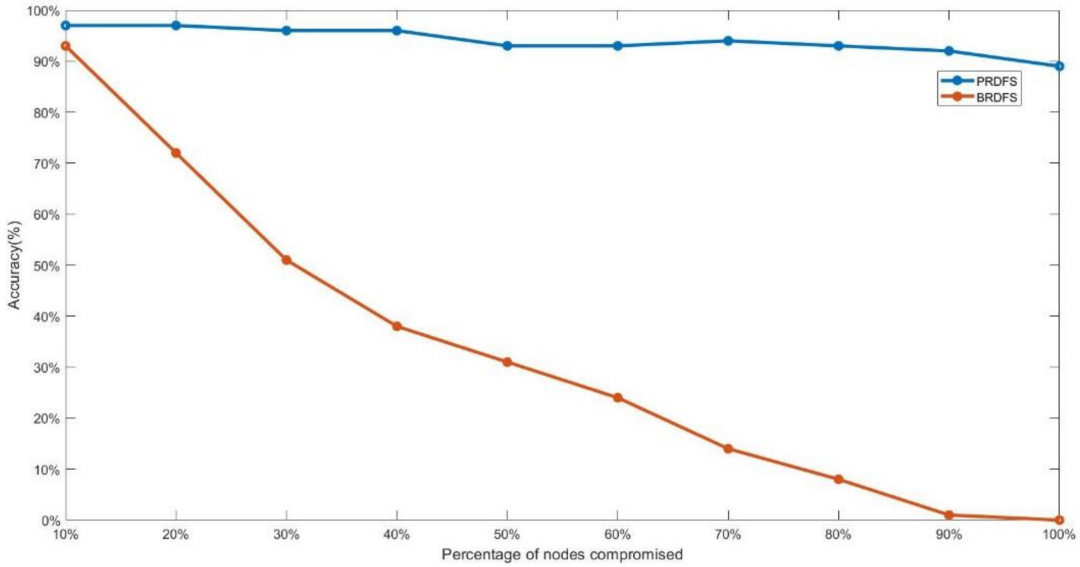

Figure 8. A comparison of accuracy between PRDFS and BRDFS for classifying normal and malicious nodes. Figure 8 depicts the node classification accuracy of PDRFS and BRDFS under the varying percentage of malicious sensor nodes. The X-axis represents the percentages of sensor nodes that are malicious in the simulation trial. The Y-axis represents the mean accuracy of classifying the sensor nodes into one of the two classes (malicious and normal) over the 1000-time steps. The classification is done at each time step. The blue line shows the classification accuracy of PRDFS under different percentages of malicious nodes, whereas the orange line shows the mean classification accuracy of BRDFS under different percentages of malicious nodes. PRDFS achieves high malicious node detection accuracy. The accuracy of PRDFS is almost always over 90%. The orange line shows a dramatic fall in the accuracy of BRDFS as the number of malicious nodes increases. Its accuracy varies from over 90% to 0%.

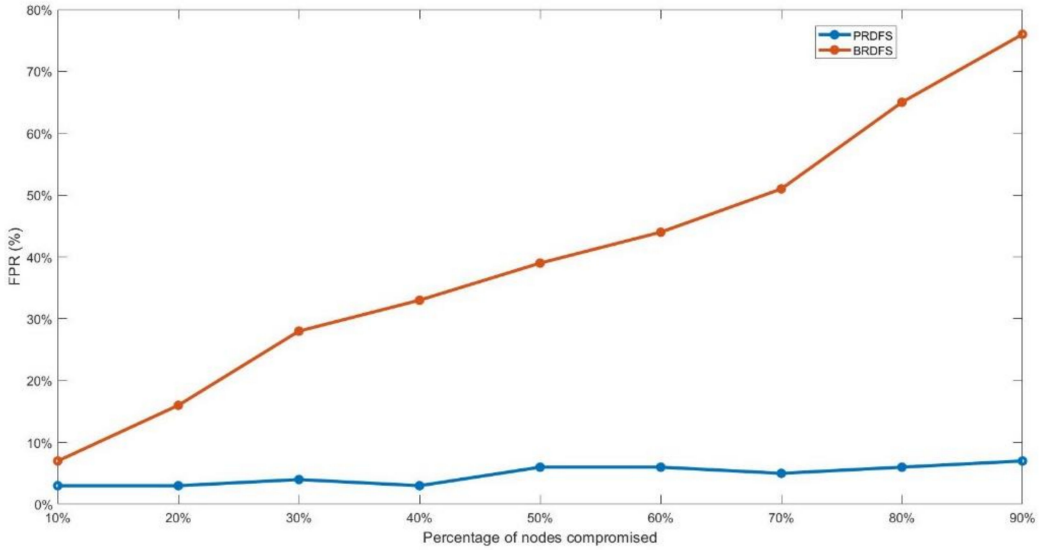

Figure 9. A comparison of False Positive Rates (FPR) for PRDFS and BRDFS for detecting malicious nodes. Figure 9 shows the false positive rates (FPR) for PRDFS and BRDFS when detecting malicious nodes. The X-axis represents the percentages of sensor nodes that were malicious in the simulation trial. The Y-axis represents the false positive rates (FPR). The blue line shows the FPR of PRDFS under

different percentages of malicious nodes whereas the orange line shows the FPR of BRDFS under different percentages of malicious nodes. The FPR in PRDFS never exceeds 9%. The FPR of BRDFS rises sharply with the percentage of malicious nodes. The FPR of BRDFS varies from less than 10% to around 75%.

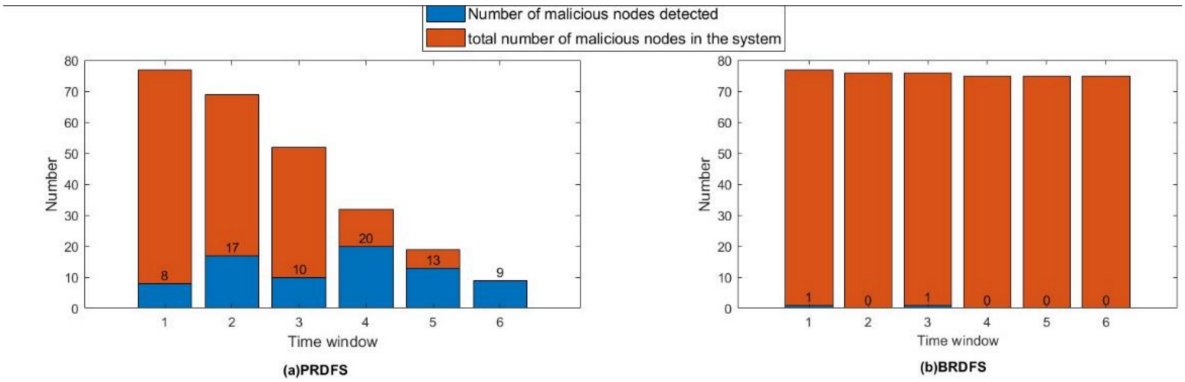

Figure 10. The distribution of detected malicious nodes by PRDFS and BRDFS over time. (**a**) PRDFS and (**b**) BRDFS showing the distributions of the number of malicious nodes detected by PRDFS and BRDFS, respectively, in contrast to total number of malicious nodes in the system in different time windows in all games. The length of each time window is equal to five-time steps. The X-axis represents the time window, and the Y-axis represents the number of malicious nodes. In one game, the total number of nodes is 14. Ten games out of all 11 games contain malicious nodes. The total number of nodes in ten games together is 140, among which 77 nodes are malicious. Nodes are compromised at the beginning of the game. If a malicious node is detected, it is converted to benign nodes. If an algorithm detects a higher number of malicious nodes early in the games, there will be a lower total number of malicious nodes later. We add up malicious nodes detected in all games at each time window to get the total number of malicious nodes at the given time window. The orange part of each stacked bar represents the total number of malicious nodes undetected at the beginning of each time window, whereas the blue part represents the total number of malicious nodes detected in that time window. PRDFS detects the largest number of malicious nodes at the fourth time window. BRDFS appears to have a shorter time window for detecting malicious nodes than PRDFS. BRDFS have not detected any malicious nodes after the third time window.

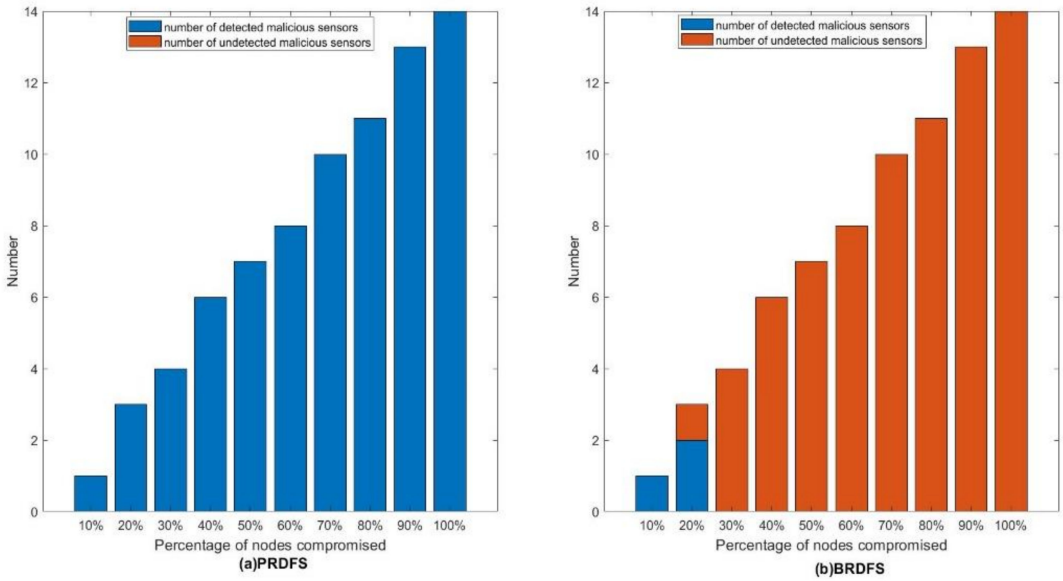

Figure 11. Number of malicious nodes detected vs. number of malicious nodes undetected by PRDFS and BRDFS. (**a**) PRDFS, and (**b**) BRDFS. Each bar in the subplots '(**a**) PRDFS' and '(**b**) BRDFS' shows the number of malicious nodes detected and the number of malicious nodes undetected by PRDFSand

BRDFS, respectively, under a given percentage of malicious nodes. In both subplots, the X-axis represents the percentage of nodes compromised, whereas the Y-axis represents the number of nodes. The blue part of the bar indicates the number of detected malicious nodes. The orange part of the bar indicates the number of undetected malicious nodes. In subplot '(**a**)PRDF', there is no orange part in any bar. This is because PRDFS has detected all the malicious nodes under all percentages of malicious nodes. In subplot '(**b**)BRDFS', on the contrary, all bars are completely orange, except for the first two bars. A completely orange bar indicates that no malicious node is detected by BRDFS under the given percentage of malicious nodes. BRDFS identifies all malicious nodes when the percentage of malicious nodes is 10%. It can detect most of the malicious nodes under attacks with 20% malicious nodes.

5.2.2. Further Observations

We attribute the above observations to the following characteristics of PRDFS and BRDFS:

- PRDFS uses a supervised intrusion detection system (SIDS) for detecting malicious nodes.
  - ➢ SIDS learns from the input and output data sets that are contamination-free since they are generated through the game theoretic approach. It employs a supervised learning algorithm (GRNN) to distinguish between malicious and normal nodes.
  - ➢ It can detect malicious nodes even when all nodes are malicious because it compares the present behaviors of the nodes in respect to dynamic spatio-temporal patterns of trustworthiness and the skewness of residual distribution with previous correct behaviors.
- BRDFS uses an unsupervised anomaly-based intrusion detection system (AIDS) for detecting malicious nodes.

➢ AIDS learn from unlabeled real data that are contaminated with many undetected malicious nodes. Hence, its learning is not very effective.

➢ AIDS learn under the assumption that statistically rare or atypical behaviors are abnormal and are possible signs of maliciousness, whereas typical or common behaviors are normal. Hence, AIDS can only detect malicious nodes when the percentage of malicious nodes is low, and the malicious nodes are operating for a short period of time where it is not possible to inject a significant number of malicious instances in the data set. However, stealthy malicious nodes are highly difficult to detect at very early stage.

➢ The skewness detector and the temporal collusion detection scheme of CAD uses confidence interval-based anomaly detection. If a node's behavioral traits fall outside the confidence interval, the node is detected as malicious. However, such a strategy is ineffective against collusion attacks. In collusion attacks, the adversary takes control of multiple sensors and changes the readings of these sensors to the desired upward or downward direction in such a way that the malicious nodes always lie within the confidence intervals of correlation coefficients and skewness coefficients. One of the major challenges in time series forecasting is that the distribution of the time series changes over time. Hence, the forecasting model needs to be adapted to the most recent data—the adversaries take advantage of this situation. Returning the upper or lower extreme of the acceptable range as the readings of the compromised sensors, they can gradually move the forecasting model in the wrong direction without getting caught. Since the forecasting model is evolving towards the false data, over time, the non-compromised sensors are marked as compromised one after another and will lose all influence over the model's evolution, thereby accelerating the compromise rate of the entire forecasting system.

➢ The spatial collusion detector of the CAD in BRDFS uses unsupervised clustering to separate malicious nodes. These clustering algorithms are based on arbitrary rules. Thus, it is natural that its accuracy is not as good as PRDFS.

5.3. The Comparative Performance of PRDFS and BRDFS in Estimating the Most Trustworthy Measurement of the Current Aggregate Demand $\left(\overrightarrow{D(t)}\right)$

5.3.1. Major Findings

- The demand forecast resilience is directly correlated to the accuracy in estimating current aggregate demand $\left(\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}\right)$ that acts as the input signal while forecasting the demand time series (Figures 7 and 12).
- Since PRDFS detects and recovers all compromised nodes at the very early stages of games (within first 30-time steps) (Figure 10), the mean accuracy of estimating $\left(\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}\right)$ in PRDFS is almost constant to around 98% over the time frame of the game (i.e., 1000-time steps) regardless of the percentage of the nodes compromised by the adversary (Figure 12).
- During the first 40-time steps of games, while the attack is still occurring or just ended, the mean $\left(\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}\right)$ estimation accuracy in PRDFS decreases from around 98% to 83% as the percentage of malicious nodes increases. In contrast, the mean $\left(\overrightarrow{\boldsymbol{D}(\boldsymbol{t})}\right)$ estimation accuracy in BRDFS falls sharply over the simulation period (i.e., 1000 time-steps) as the percentage of malicious nodes increases.

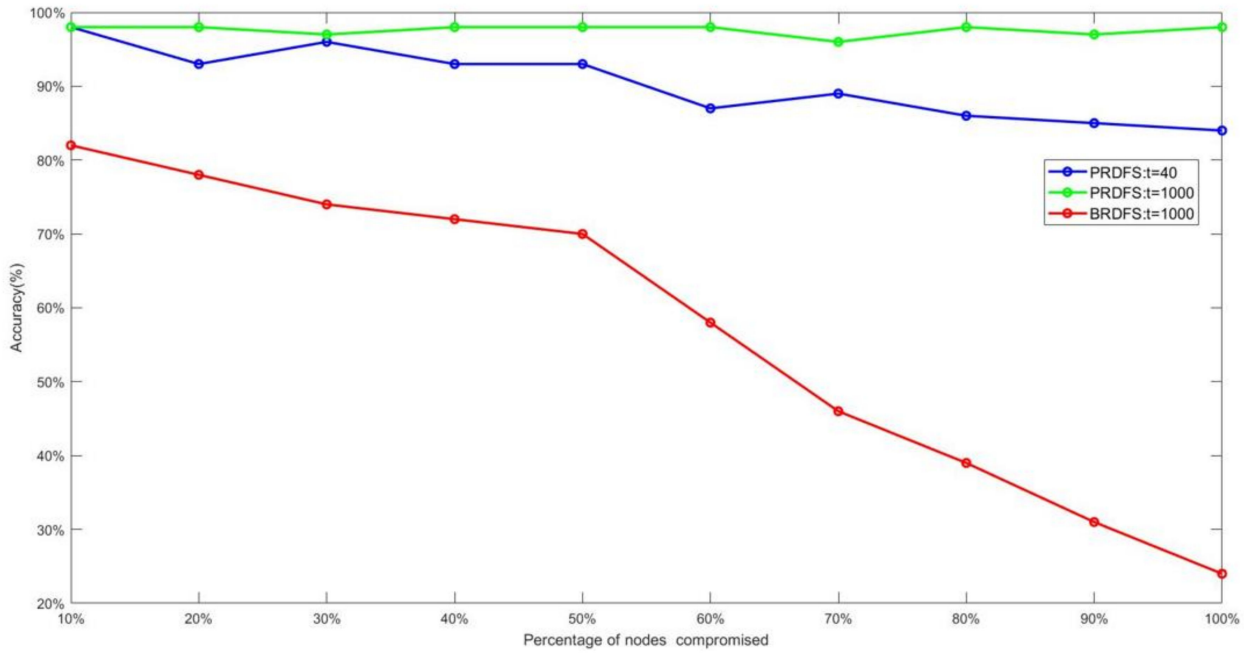

Figure 12. A comparison of accuracy between PRDFS and BRDFS for estimating current aggregate demand. Figure 12 depicts the prediction accuracy of current aggregate demands in PDRFS and BRDFS under the varying percentage of malicious sensor nodes. The accuracy is estimated based on the difference between the most trustworthy measurement of the current aggregate demand predicted by the TMS and the actual current aggregate demand (which is the simple average of the actual current measurements of the sensors). The X-axis represents the percentages of sensor nodes that were malicious in the simulation trial. The Y-axis represents the accuracy. The accuracy is estimated based on MAAPE. The green and blue lines plot the mean accuracy of PRDFS in respect to the mean current aggregate demand over 1000-time steps and over the first 40-time steps, respectively, under different percentages of malicious nodes. The red line plots the mean accuracy of BRDFS in terms of the mean current aggregate demand over 1000-time steps under different percentages of malicious nodes. The

green line is almost flat to X-axis, indicating that the mean accuracy of mean current aggregate demand over 1000-time steps predicted by (the TMS of) PRDFS is not sensitive to the percentage of malicious nodes in the system. The blue line drops gradually as the percentage of almost flat to X-axis, indicating that the mean accuracy of mean current aggregate demand over 1000-time steps predicted by (the TMS of) PRDFS is not sensitive to the percentage of malicious nodes in the system. The blue line drops gradually as the percentage of malicious nodes increases, indicating that the mean accuracy of mean current aggregate demand over the first 40-time steps predicted by PRDFS is mildly sensitive to the percentage of malicious nodes in the system. The red line falls at an accelerated rate indicating that the mean accuracy of mean current aggregate demand over 1000-time steps predicted by BRDFS is highly sensitive to the percentage of malicious nodes in the system.

5.3.2. Further Observations

The following strategy differences between the TMS of PRDFS and that of BRDFS may explain their striking differences in $\left(\overrightarrow{D(t)}\right)$ estimation accuracy in the face of an attack:

- The TMS of PRDFS considers the following three factors in determining the trustworthiness of a sensor node at a time distant t: (1) the trustworthiness of the node, (2) the similarity of the sensor's reading with the other sensors' readings at the same time step, and (3) the similarity of the sensor's measurement at time step t with its past measurements, but in contrast, the TMS of BRDFS uses only the first two factors.
- The TMS of BRDFS updates the reputation score of a node at the end of a time step t after determining the most trustworthy measurement at that time step. In the presence of on-off attacks, the normal behavior of a node at the last few time steps does not guarantee normal behavior from the node at the next time step. For this reason, the TMS of PRDFS predicts the trust score of a node at the beginning of a time step t (before determining the most trustworthy measurement at that time step) using GRNN, which is a powerful pattern recognition algorithm capable of learning local patterns.
- In BRDFS, the reputation score of a node at time step t is a linear function of its long-term reputation and its intermediate reputation score computed at time step t. In contrast, the TMS of PRDFS uses a powerful fuzzy clustering-based non-linear algorithm (GRNN) to predict the trust score of a node at time step t based on its past trustworthiness scores.
- The TMS of BRDFS sets the reputation score of a node at time step t to the estimated expected (i.e., mean) reputation score of the node at time step t. In stealthy on-off attacks, where at each time step the malicious nodes randomly choose whether to act maliciously or not, the variance of their reputation is an important indicator of their conditions. Hence, the point estimate of the expected reputation score is not very reliable. To overcome this problem, the TMS of PRDFS sets the trust score of a node to the lower bound of the predicted distribution of the plausible trust scores of the node at time step t

5.4. The Comparative Performance of PRDFS and BRDFS in Terms of Demand Forecasting when No Attack Is Underway

5.4.1. Major Findings

- PRDFS outperforms BRDFS in terms of demand forecasting accuracy by 15% or more (Figure 13).

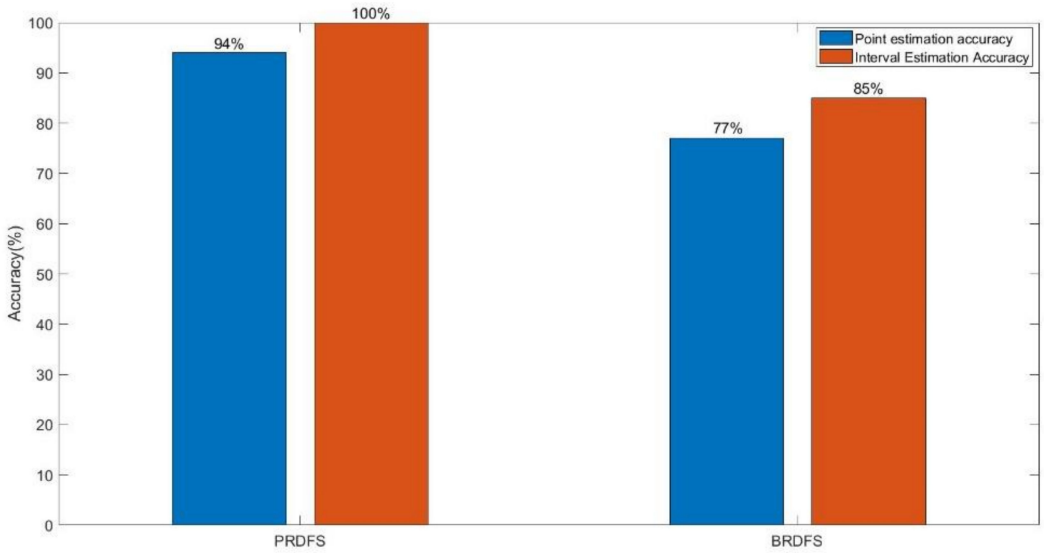

Figure 13. Comparison of demand forecasting performance between PRDFS and BRDFS when no attack is present. Figure 13 shows the demand forecasting accuracy of PRDFS and BRDFS when no attack takes place. The Y-axis of the figure represents the accuracy. The first two bars from the left are for PRDFS and the second two bars are for BRDFS. Blue bars represent the point forecasting accuracy, whereas the orange bars represent the interval forecasting accuracy. The figure shows that PRDFS outperforms BRDFS in both point and interval forecasting accuracy.

5.4.2. Possible Explanations

We attribute the finding to the robustness of PRDFS against non-stationarity and high dimensionality. To deal with the non-stationarity in the demand time series, the DFS of PRDFS applies wavelet multiresolution analysis on the raw demand time series and decomposes the original input time series into multiple component time series with a simpler structure. Each component time series contains the information content of the original demand time series at a particular level or scale. Our study suggests that the wavelet analysis eliminates the requirement of selecting optimal seasonal lagged predictor variables (i.e., non-consecutive lagged variables) from the component time series for the demand forecasting model, which are highly difficult to identify. In the process, it also reduces the dimensionality of the feature space.

- The curse of dimensionality hits particularly hard on time series forecasting models of high or even moderate dimensional input data, since they use lagged variables as predictor variables and the lagged variables are highly correlated with each other. Hence, to reduce the dimensionality further, the DFS of PRDFS uses a hierarchical ensemble model where each member CNN is trained with only a small number of non-seasonal lagged variables. Ensemble members are trained sequentially. The first ensemble member is trained on the original demand time series. Each of the remaining ensemble members is trained on the residual series of the previous trained ensemble member. Additionally, the DFS of PRDFS uses CNNs as ensemble members. Our study suggests that CNN is more robust to the high dimensionality compared to other existing learning algorithms.
- The DFS of BRDFS does not have any such effective strategy to deal with non-stationarity and high dimensionality. It just uses a single LSTM regression network to forecast the demand.

6. Conclusions

In this paper, we propose a novel resilient cyber-physical demand forecasting system (CDFS) called PRDFS that is resilient to false data injection attacks (FDIAs) since it can distinguish a natural change in the dynamics of demand over time, and a change due to the

adversary's attack. To achieve this capability, it uses a signature-based intrusion detection system (SIDS) of malicious nodes and normal nodes. SIDS is trained on labeled training data set using a supervised learning algorithm (GRNN). To automatically generate the training dataset for SIDS with all possible attack paths (defined in the spaces of assessed node trustworthiness and the skewness of residuals of forecasting models for individual sensor time series), an ethical hacking system (EHS) is run. EHS plays repeated attacker-defender games with the defense system (SIDS and TMS) of PRDFS. The training data of SIDS are continuously automatically generated by these games. This is the main novelty of PRDFS.

PRDFS uses a trust management system (TMS) to determine the most trustworthy measurement at time step t from noisy sensor measurements for the aggregate demand time series. It considers the following information while determining the most trustworthy measurement at time step t: spatial data similarity, temporal data similarity, and the minimum expected trustworthiness of the corresponding sensor node at time step t. TMS employs GRNNs to predict the minimum expected trust score of each sensor based on the past trustworthiness scores of its measurements.

The demand forecasting system (DFS) of PRDFS uses the lagged values of aggregate demand time series to make out-of-sample forecasts. Our study suggests that among all existing approaches for dealing with the non-stationarity problem, the wavelet multiresolution decomposition approach is the most effective. Our DFS decomposes the time series into multiple component time series using wavelet transform. To avoid the curse of dimensionality, the proposed DFS uses a hierarchical ensemble of convolutional neural networks (CNNs) where each ensemble member is trained with a relatively small number of non-seasonal lag values of the component time series. Each ensemble member is trained sequentially. The first ensemble member is trained on the component time series derived from the original demand time series. Each of the remaining members is trained on the component time series derived from the residual series of the previous ensemble member. The number of ensemble members depends on how many members are required to make the final residual series of the ensemble a white noise series. This is another novelty of the proposed PRDFS.

For comparison purposes, we implement a hybrid resilient demand forecasting system that incorporates the best modules of existing demand forecasting systems. We call this resilient demand forecasting system BRDFS. We compare the performance of PRDFS and BRDFS against stealthy FDIAs. The data streams of all sensors are generated synthetically. Both were tested in 11 simulation setups with different percentages of malicious nodes. The performance of our algorithm is highly encouraging. It detects all malicious nodes within the first 30-time steps and it has low false positive rates (FPRs), never exceeding 9% regardless of the percentage of malicious nodes. Its demand forecasting accuracy never drops below 80% in the face of attacks.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app121910093/s1, The following supporting information is attached in the zip folder: (1) the source code of the proposed algorithm 'PRDFS'; (2) the datasets used in this study; and (3) ReadMe: explains how to run the source code, and includes a brief overview of the datasets used in the study; (4) the pseudocode of PRDFS

Author Contributions: Conceptualization, I.G.; Data curation, G.E. and S.L.; Formal analysis, I.G.; Funding acquisition, C.M.; Methodology, I.G.; Project administration, G.E. and C.M.; Supervision, G.E.; Validation, S.L.; Writing—original draft, I.G. and G.E.; Writing—review and editing, C.M. and S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the I-TRACE programme, 10.13039/501100006041-Innovate UK (Grant Number: 105590).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gudova, I.A.; Frau, O. Critical infrastructure in the era of IOT and automation. In *Proceedings of the International Scientific Conferences*; "Strategies XXI", Supple; Command and Staff Faculty: Bucharest, Romania, 2020.
2. Muralidhar, N.; Muthiah, S.; Sharma, R.; Ramakrishnan, N. Multivariate long-term state forecasting in cyber-physical systems: A sequence to sequence approach. In Proceedings of the 2019 IEEE International Conference on Big Data, Los Angeles, CA, USA, 9–12 December 2019; pp. 543–552. [CrossRef]
3. Chekola, E.G.; Ochoa, M.; Chattopadhyay, S. SCOPE: Secure compiling of PLC in cyber-physical systems. *Int. J. Crit. Infrastruct. Prot.* **2021**, *33*, 100431. [CrossRef]
4. Rogers, R.; Apech, E.; Richardson, C.J. Resilience of the Internet of Things (IoT) from an information assurance (IA) perspective. In Proceedings of the 2016 10th International Conference on Software, Knowledge, Information Management & Applications (SKIMA), Chengdu, China, 15–17 December 2016. [CrossRef]
5. Lee, J.; Bagheri, B. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manuf. Lett.* **2015**, *3*, 18–23. [CrossRef]
6. Asshibani, Y.; Mahmoud, Q.H. Cyber physical systems security: Analysis, challenges and solutions. *Comput. Secur.* **2017**, *68*, 81–97.
7. Sethi, P.; Sarangi, S. Internet of Things: Architectures, protocols, and applications. *J. Electr. Comput. Eng.* **2017**, *2017*, 9324035. [CrossRef]
8. Ghalehkhondabi, I.; Ardjmand, E.; Young, W.A.; Weckman, G.R. Water demand forecasting: Review of soft computing methods. *Environ. Monit. Assess.* **2017**, *189*, 313. [CrossRef]
9. Zhou, X.; Li, Y.; Barreto, C.A.; Li, J.; Volgyesi, P.; Neema, H.; Koutsoukos, X. Evaluating resilience of grid load predictions under stealthy adversarial attacks. In Proceedings of the 2019 Resilience Week (RWS), San Antonio, TX, USA, 4–7 November 2019; pp. 206–212. [CrossRef]
10. Hoque, M.E.; Thavaneswaran, A.; Appadoo, S.S.; Thulasiram, R.K.; Banitalebi, B. A novel dynamic demand forecasting model for resilient supply chains using machine learning. In Proceedings of the 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 12–16 July 2021. [CrossRef]
11. Karthik, N.; Ananthanarayana, V.S. Data trustworthiness in wireless sensor networks. In Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016. [CrossRef]
12. Agarwal, V.; Pal, S.; Sharma, N.; Sethi, V. Identification of defective nodes in cyber-physical systems. In Proceedings of the 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Delhi, India, 10–13 December 2020. [CrossRef]
13. El-Rewini, Z.; Sadatssharan, K.; Sugunaraj, N.; Selvaraj, D.F.; Plathottam, S.J.; Ranganathan, P. Cybersecurity attacks in vehicular sensors. *IEEE Sens. J.* **2020**, *20*, 13752–13767. [CrossRef]
14. Jeba, S.V.; Paramasivan, B. Faalse data injection attack and its countermeasures in wireless sensor networks. *Eur. J. Sci. Res.* **2012**, *82*, 248–257.
15. Tian, M.; Dong, Z.; Wang, X. Analysis of false data injection attackss in power systems: A dynamic Bayesian game-theoretic approach. *ISA Transsactions* **2021**, *115*, 108–123. [CrossRef]
16. Kumari, A.; Tanwar, S. RAKSHAK: Resilient and scalable demand response management scheme for smart grid systems. In Proceedings of the 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 28–29 January 2021; pp. 309–314. [CrossRef]
17. Barreto, C.; Koutsoukos, X. Design of load forecast systems resilient against cyber-attacks. In *Decision and Game Theory for Security. Alpcan, T., Vorobeychik, Y., Baras, J., Dan, G., Eds.*; Springer: Cham, Switzerland, 2019. [CrossRef]
18. Babar, M.; Tariq, M.U.; Jan, M.A. Secure and resilient demand side management engine using machine learning for IoT-enabled smart grid. *Sustain. Cities Soc.* **2020**, *62*, 102370. [CrossRef]
19. Chen, Q.; Zhang, C.; Zhang, S. Detection Models of Collusion Attacks. In *Secure Transaction Protocol Analysis*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5111. [CrossRef]
20. Karthik, N.; Ananthanarayana, V.S. A hybrid trust management scheme for wireless sensor networks. *Wirel. Pers. Commun.* **2017**, *97*, 5137–5170. [CrossRef]
21. Labraoui, N.; Gueroui, M.; Sekhri, L. On-off attacks mitigation against trust systems in wireless sensor networks. In *Computer Science and Its Applications*; Amine, A., Bellatreche, L., Elberrichi, Z., Neuhold, E., Wrembel, R., Eds.; IFIP Advances in Information and Communication Technology; Springer: Cham, Switzerland, 2015; Volume 456. [CrossRef]
22. Lv, Z.; Han, Y.; Singh, A.K.; Manogaran, G.; Lv, H. Trustworthiness in industrial IoT systems based on artificial intelligence. *IEEE Trans. Ind. Inform.* **2021**, *17*, 1496–1504. [CrossRef]
23. Hassan, S.; Khossravi, A.; Jaafar, J. Examining performance of aggregation algorithms for neural network-based electricity demand forecasting. *Electr. Power Energy Syst.* **2015**, *64*, 1098–1105. [CrossRef]
24. Gheitasi, K.; Lucia, W. Undetectable Finite-Time Covert Attack on Constrained Cyber-Physical Systems. *IEEE Trans. Control. Netw. Syst.* **2022**, *9*, 1040–1048. [CrossRef]

25. Ozer, O.; Zheng, Y. Trust and Trustworthiness. In *The Handbook of Behavioral Operations*, 1st ed.; Donohue, K., Katok, E., Leider, S., Eds.; Wiley Online Library: Hoboken, NJ, USA, 2018; pp. 489–523.
26. Lopez, J.; Roman, R.; Agudo, I.; Fernandez-Gago, C. Trust management systems for wireless sensor networks:best practices. *Comput. Commun.* **2010**, *33*, 1086–1093. [CrossRef]
27. Mohammadi, V.; Rahmani, A.M.; Darwesh, A.M.; Sahafi, A. Trust-based recommendation systems in internet of things: A systematic literature review. *Hum.-Cenric Comput. Inf. Sci.* **2019**, *9*, 21. [CrossRef]
28. Ryutov, T.; Neuman, C. Trust based approach for improving data reliability in industrial sensor networks. In *IFIP International Federation for Information Processing*; Etalle, S., Marsh, S., Eds.; Springer: Boston, MA, USA, 2007; Volume 238, pp. 154–196. [CrossRef]
29. Momani, M.; Challa, S.; Al-Hmouz, R. Bayesian fusion algorithm for inferring trust in wireless sensor networks. *J. Netw.* **2010**, *5*, 815–822. [CrossRef]
30. Marinenkov, E.; Chuprov, S.; Viksnin, I.; Kim, I. Empirical study on trust, reputation, and game theory approach to secure communication in a group of unmanned vehicles. In Proceedings of the MICSECS, Saint Petersburg, Russia, 12–13 December 2019.
31. Lim, H.-S.; Moon, Y.-S.; Bertino, E. Provenance-based trustworthiness assessment in sensor networks. In Proceedings of the Seventh International Workshop on Data Management for Sensor Networks, Singapore, 13 September 2010; pp. 2–7. [CrossRef]
32. Lotufo, A.D.P.; Minussi, C.R. Electric power systems load forecasting: A survey. In Proceedings of the International Conference on Electric Power Engineering, Power Tech, Budapest, Budapest, Hungary, 29 August–2 September 1999. [CrossRef]
33. Fallah, S.N.; Deo, R.C.; Shojafar, M.; Conti, M.; Shamshirband, S. Computational intelligence approaches for energy load forecasting in smart energy management grids: State of the art, future challenges, and research directions. *Energies* **2018**, *11*, 596. [CrossRef]
34. Bose, J.-H.; Flunkert, V.; Gasthaus, J.; Januschowski, T.; Lange, D.; Salinas, D.; Schelter, S.; Seeger, M.; Wang, Y. Probabilistic demand forecasting at scale. *Proc. VLDB Endow.* **2017**, *10*, 1694–1705. [CrossRef]
35. Salles, R.; Belloze, K.; Porto, F.; Gonzalez, P.H.; Ogasawara, E. Nonstationary time series transformation methods: An experimental review. *Knowl.-Based Syst.* **2019**, *164*, 274–291. [CrossRef]
36. Panapakidis, I.P.; Dagoumas, A.S. Day-ahead natural gas demand forecasting based on the combination of wavelet transform and ANFIS/genetic algorithm/neural network model. *Energy* **2017**, *118*, 231–245. [CrossRef]
37. Hu, Y.; Li, H.; Yang, H.; Sun, Y.; Sun, L.; Wang, Z. Detecting stealthy attacks against industrial control systems based on residual skewness analysis. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 74. [CrossRef]
38. Bhuiyan, M.Z.A.; Wu, J. Collussion attack detection in networked systems. In Proceedings of the 2016 IEEE 14th Intl Conf on Dependable, Automatic and Secure Computing, Aukland, New Zealand, 8–12 August 2016; pp. 286–293. [CrossRef]
39. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. In Proceedings of the 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 21–23 August 2017. [CrossRef]
40. Specht, D.F. A general regression neural network. *IEEE Trans. Neural Netw.* **1991**, *2*, 568–576. [CrossRef] [PubMed]
41. Zhang, D. Wavelet Transform. In *Fundamentals of Image Data Mining*; Texts in Computer Science; Springer: Cham, Switzerland, 2019. [CrossRef]
42. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
43. Mishra, M.K.; Murari, K.; Parida, S.K. Demand-side management and its impact on utility and consumers through a game theoretic approach. *Int. J. Electr. Power Energy Syst.* **2022**, *140*, 107995. [CrossRef]
44. Fu, W.; Chien, C-F. UNISON data-driven intermittent demand forecast framework to empower supply chain resilience and an empirical study in electronics distribution. *Comput. Ind. Eng.* **2019**, *135*, 940–949. [CrossRef]
45. Kim, S.; Kim, H. A new metric of absolute percentage error for intermittent demand forecasts. *Int. J. Forecast.* **2016**, *32*, 669–679. [CrossRef]